Receiving and Processing Messages

Alan Smith
ACTIVE SOLUTION

@alansmith www.cloudcasts.net

Overview

Receiving and Processing Messages

Demo: Receiving and Processing messages

Duplicate Detection

Demo: Duplicate Detection

Receiving and Processing Messages

Multi-threaded Message Receiving

Receive and Delete

- **Message received and deleted in one operation**
- **No option to abandon, defer, or dead-letter message**
- **At most once delivery**
  - No duplicate processing
  - Possible message loss

Peak Lock

- **Two-phase receive**
- **Message can be abandoned, deferred or dead-lettered**
- **Receiver is responsible for message completion**
- **At least once delivery**
  - No message loss
  - Possible message duplication

Message Receive Modes

Receive and Delete

Message received and deleted in one operation

No option to abandon, defer, or dead-letter message

At most once delivery

- No duplicate processing
- Possible message loss

Peak Lock

Two-phase receive

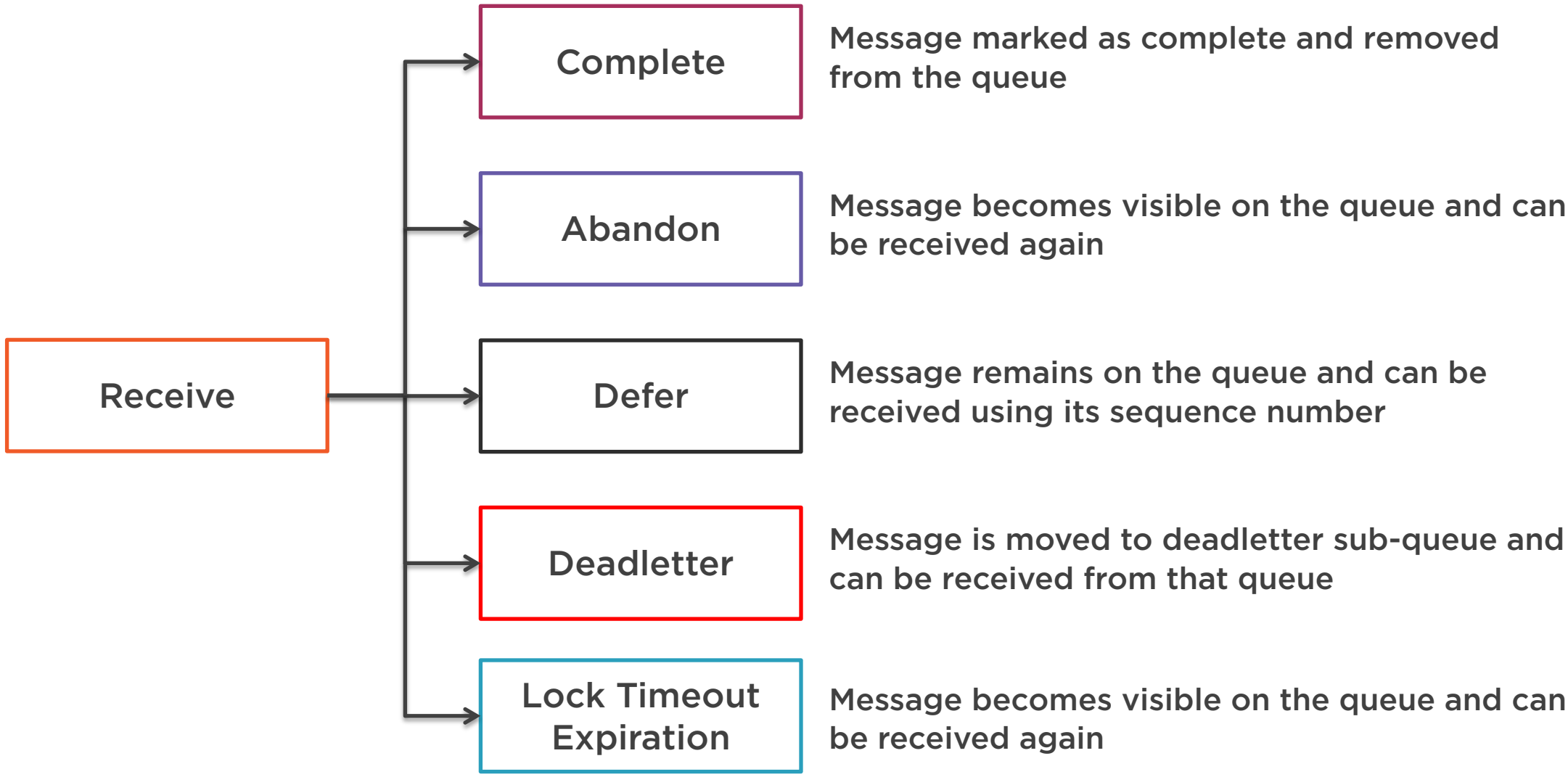
Message can be abandoned, deferred or dead-lettered

Receiver is responsible for message completion

At least once delivery

- No message loss
- Possible message duplication

Receiving Messages in Peek-lock Mode



Multi-threaded Message Receiving

Single Thread

- Easy to implement
- In-Order processing
- Slow message throughput
- Wasteful of resources

Multiple Threads

- Greatly improved message throughput
- Message order not preserved
- Challenging to implement correctly
- Has potential to flood downstream systems

Constrained Concurrency

- Limit processing to a set number of threads
- Can provide optimal throughput
- RegisterMessageHandler available in Service Bus SDK

RegisterMessageHandler

```
// Register a message handler
m_QueueClient.RegisterMessageHandler
    (HandleMessage, new MessageHandlerOptions(HandleMessageExceptions));
```

```
private static async Task HandleMessage
    (Message message, CancellationToken cancellationToken)
{
    // Process the message...

    // Complete the receive operation...
    await m_QueueClient.CompleteAsync(message.SystemProperties.LockToken);
}
```

MessageHandlerOptions

```
var options = new MessageHandlerOptions(ExceptionReceivedHandler)
{
    // We will manually complete or dead-letter messages
    AutoComplete = false,

    // Process up to 30 messages concurrently
    MaxConcurrentCalls = 30,

    // Renew message whilst processing message
    MaxAutoRenewDuration = TimeSpan.FromMinutes(10)
};
```

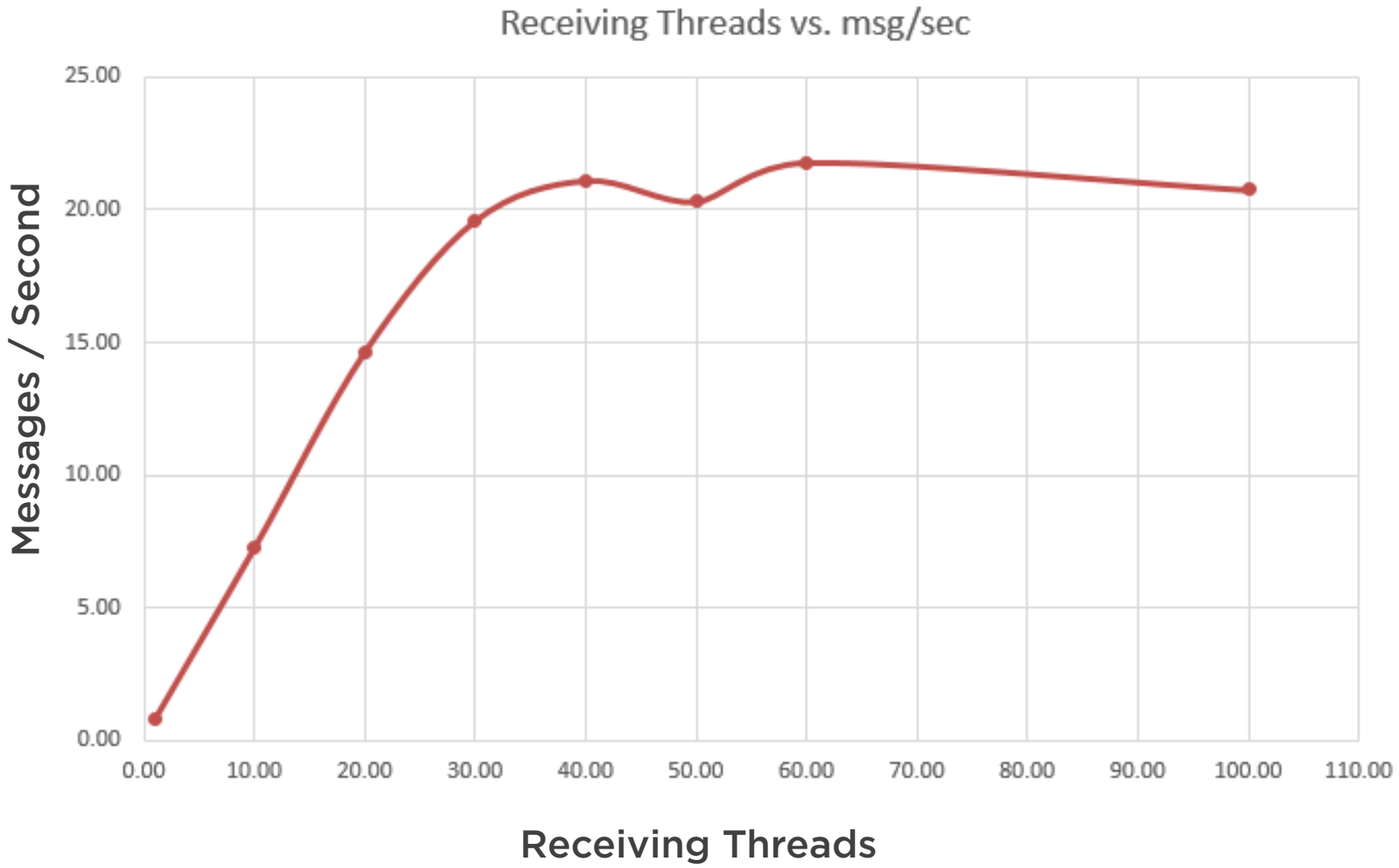
How Not to Set MaxConcurrentCalls

```
var options = new MessageHandlerOptions(ExceptionReceivedHandler)
{
    MaxConcurrentCalls = 100,
    AutoComplete = false
};
```

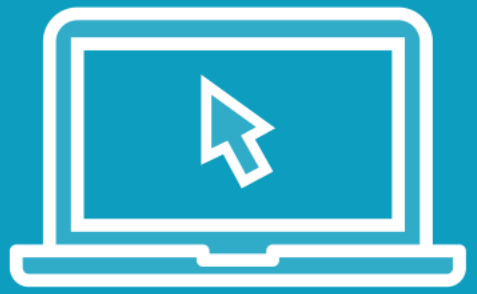
```
var options = new MessageHandlerOptions(ExceptionReceivedHandler)
{
    MaxConcurrentCalls = int.MaxValue,
    AutoComplete = false
};
```

```
var options = new MessageHandlerOptions(ExceptionReceivedHandler)
{
    MaxConcurrentCalls = DateTime.UtcNow.Millisecond + 1,
    AutoComplete = false
};
```

Analyzing Optimal Message Throughput



Demo



Receiving and Processing Messages

- Receiving messages
- Registering a message handler
- Using constrained concurrency

Duplicate Detection

Enabling Duplicate Detection

Duplicate detection can be enabled on queues and topics

- Set RequiresDuplicateDetection property to true

Enabling Duplicate Detection

Duplicate detection can be enabled on queues and topics

- Set RequiresDuplicateDetection property to true

Duplicate detection is based on MessageId property of messages

- Sending application must set this explicitly

Enabling Duplicate Detection

Duplicate detection can be enabled on queues and topics

- Set RequiresDuplicateDetection property to true

Duplicate detection is based on MessageId property of messages

- Sending application must set this explicitly

Duplicate messages are not dead-lettered

Enabling Duplicate Detection

Specify properties when creating queue

```
// Create a description for the queue.
QueueDescription rfidCheckoutQueueDescription =
    new QueueDescription(AccountDetails.QueueName)
    {
        // Enable duplicate detection with a 60 minute window
        RequiresDuplicateDetection = true,
        DuplicateDetectionHistoryTimeWindow = TimeSpan.FromMinutes(60),
};

// Create a queue based on the queue description.
await managementClient.CreateQueueAsync(rfidCheckoutQueueDescription);
```

Setting Message ID

Messages will have unique message IDs when created

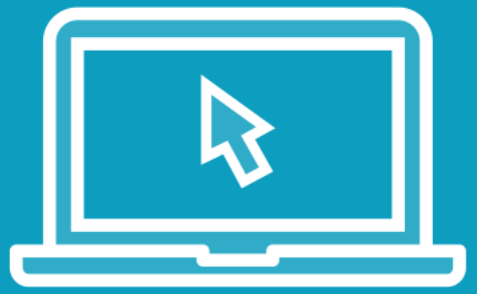
Message ID must be explicitly set before message is sent

```
// Create a new  message from the order item RFID tag.
var orderJson = JsonConvert.SerializeObject(rfidTag);
var tagReadMessage = new Message(Encoding.UTF8.GetBytes(orderJson));

// Set the message id
tagReadMessage.MessageId = rfidTag.TagId;

// Send the message
await queueClient.SendAsync(tagReadMessage);
```

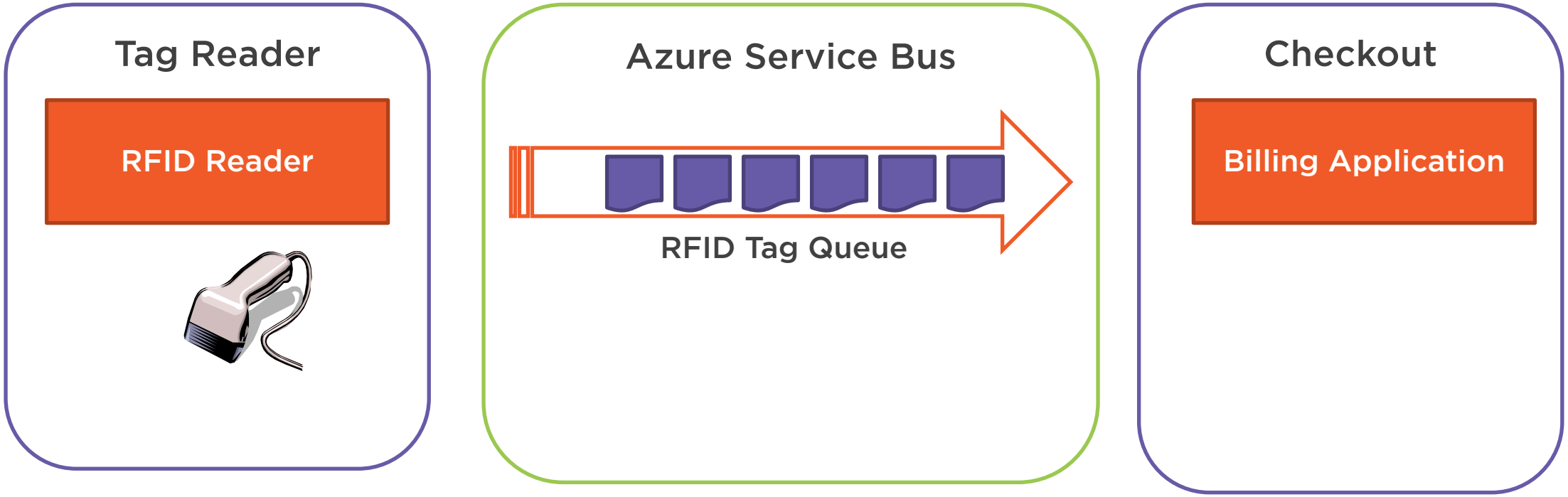
Demo



Duplicate Detection

- Enabling duplicate detection on a queue
- Ensuring duplicate detection works

Demo Scenario



Summary

Messages received in the peek-lock receive mode require a second operation

- Complete
- Abandon
- Dead-letter
- Defer

Registering a message receiver can be used as an efficient way of receiving and processing messages

Testing is required to determine the appropriate number of concurrent threads to use

Duplicate detection can be used to suppress duplicate messages

Duplicate detection is based on the MessageId property of the message