Poison Messages, Dead-Lettering & Error Handling

Alan Smith
ACTIVE SOLUTION

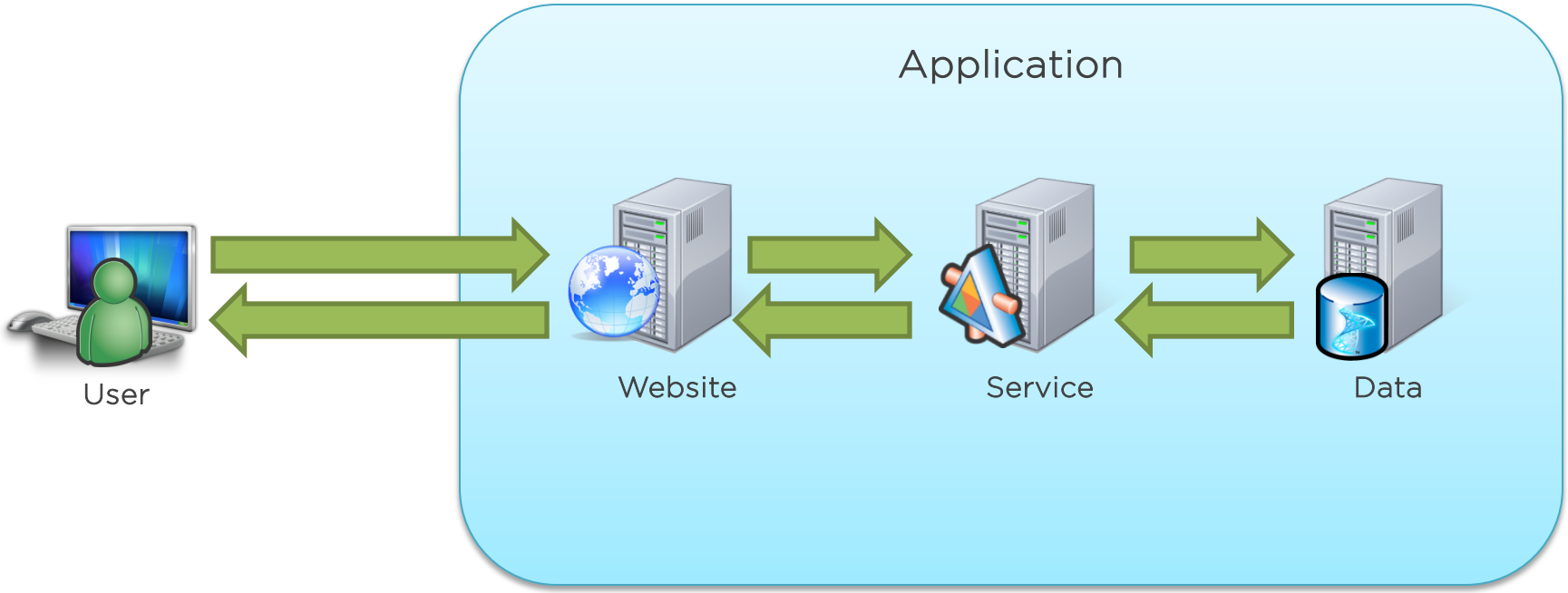
@alansmith www.cloudcasts.net

Overview

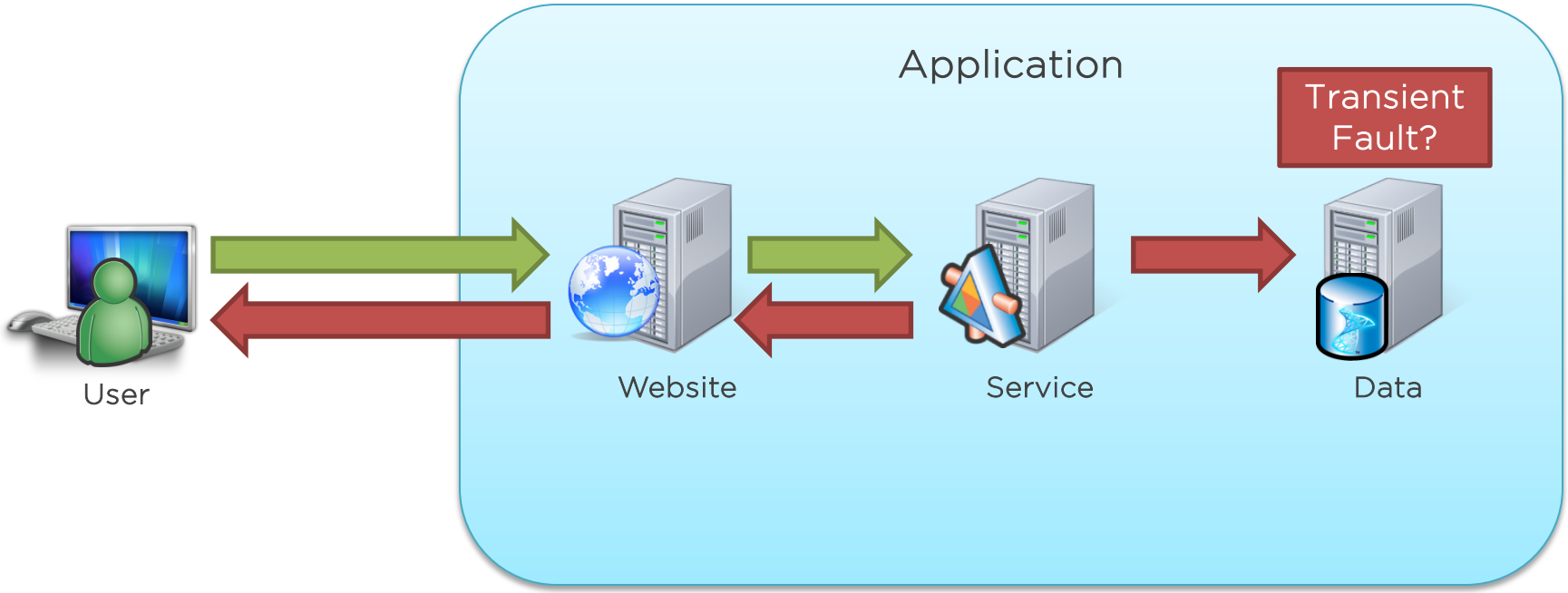
- Communication Problems
- Transient Fault Handling
- Dead-Lettering
- Demo: Dead-Lettering Messages
- Handling Poison Messages
- Demo: Handling Poison Messages

Communication Problems

Distributed Application

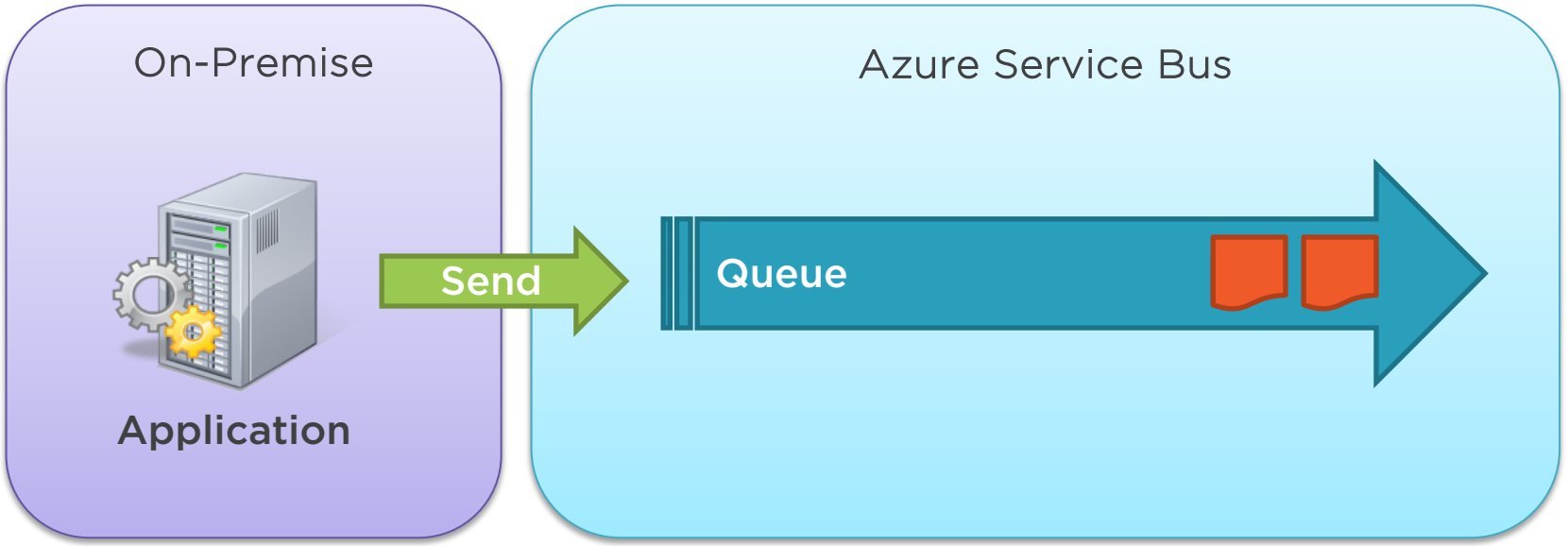


Handling Faults



Transient fault – Error that occurs due to a temporary condition

Handling Faults – Sending Messages



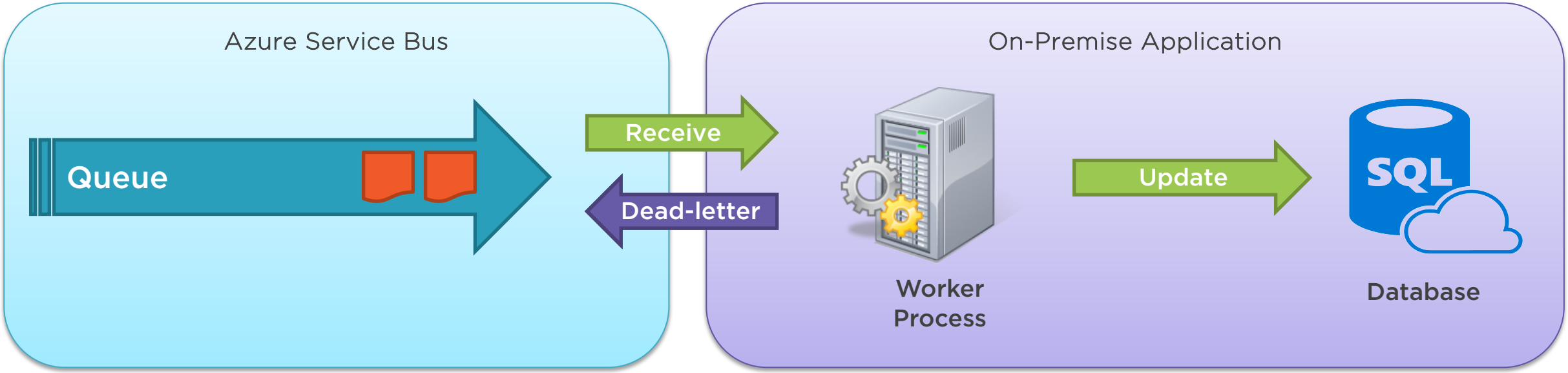
Possible Errors

- Network error
- Throttling exception
- Internal service error in service bus
- Security exception
- Endpoint not found exception

Options

- Try to resend message
- Report an error

Handling Faults – Processing Messages



Possible Errors

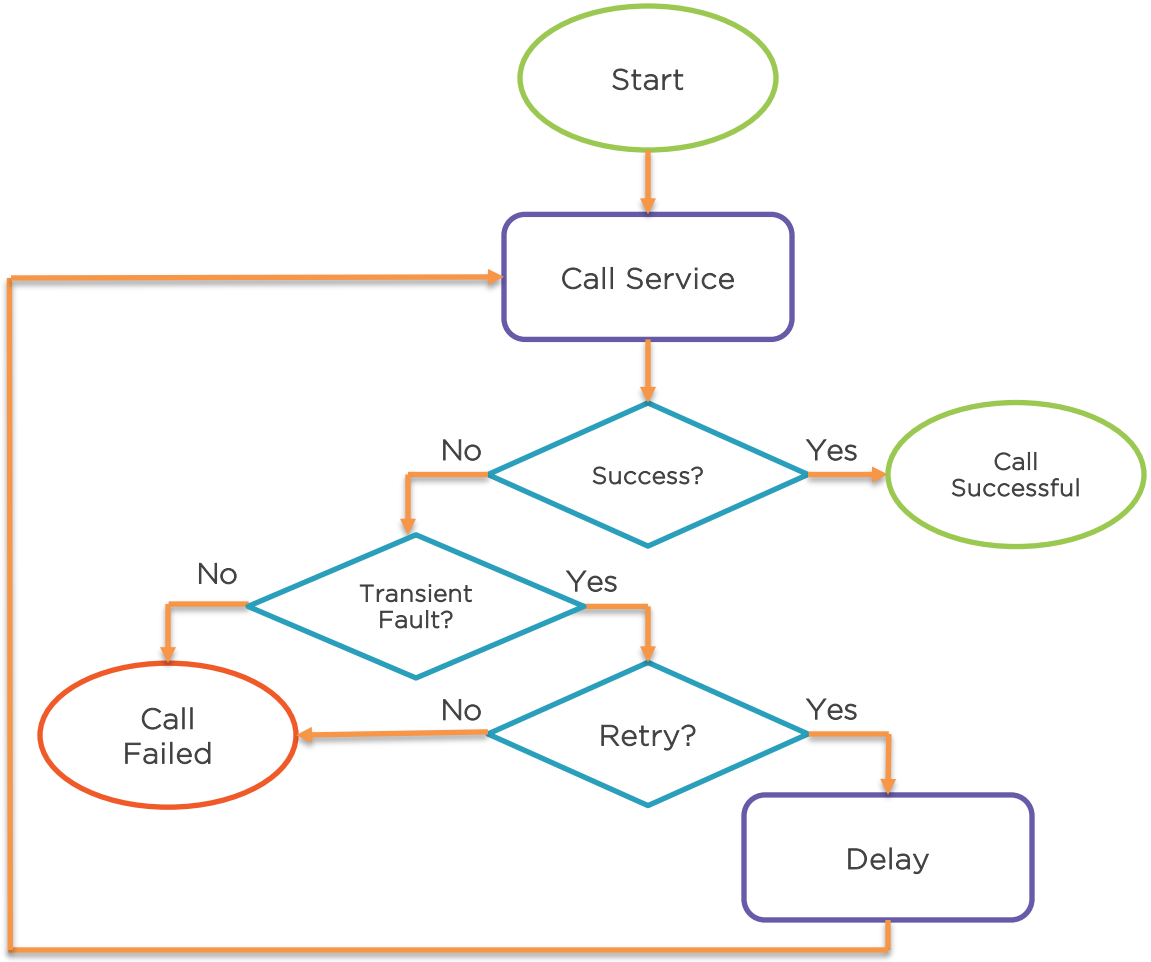
- Data integrity violation
- Security exception
- Throttling exception

Options

- Abandon message
- Dead-letter message
- Do nothing

Transient Fault Handling

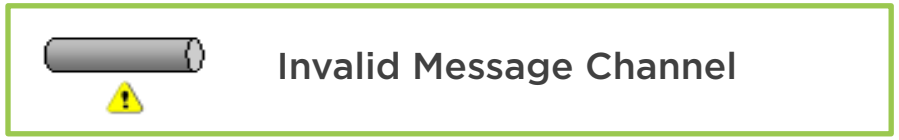
Transient Fault Handling Flowchart



Dead-lettering

Enterprise Integration Patterns

The messaging system moves messages to a dead-letter channel.



The receiving application moves messages to an invalid message channel.

http://www.enterpriseintegrationpatterns.com/

Dead-lettering Scenarios

Processing Failure

- A receiving application fails repeatedly to process a message

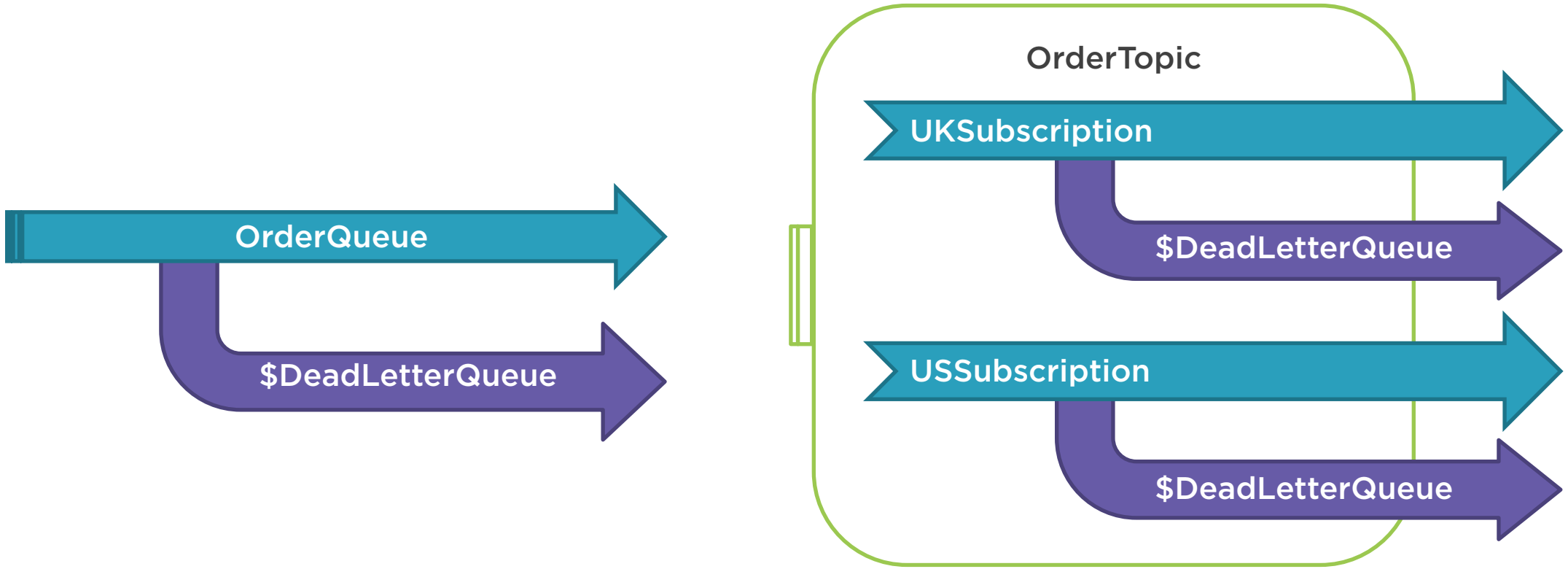
Poison Messages

- The content of a message makes it not possible to process

Expired Messages

- A message has been present in the messaging system past its expiration time

Dead-letter Sub-queues



- Queues
  - Each queue has a dead-letter sub-queue
- Topics and Subscriptions
  - Each subscription has a dead-letter sub-queue

Maximum Delivery Count

MaxDeliveryCount Property

Configurable on queues and subscriptions

Messages that repeatedly fail processing can be dead-lettered automatically

Default value is 10

However…

These dead-lettered messages contain no meaningful information as to the cause of processing failure

Maybe not the best way to handle processing failures

Consider this as a safety net

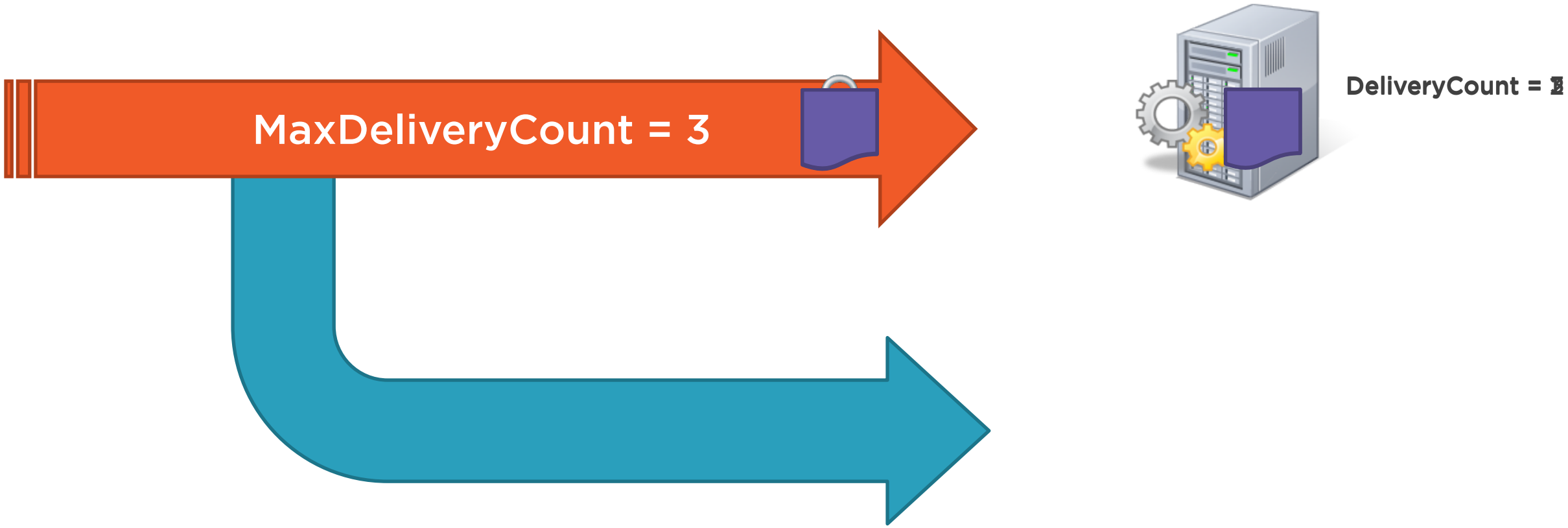
- Strive not to need it

Maximum Delivery Count

```
var subscriptionDescription = new SubscriptionDescription
    ("topic", "subscription")
{
    MaxDeliveryCount = 5
};

await managementClient.CreateSubscriptionAsync(subscriptionDescription);
```

MaxDeliveryCount Property

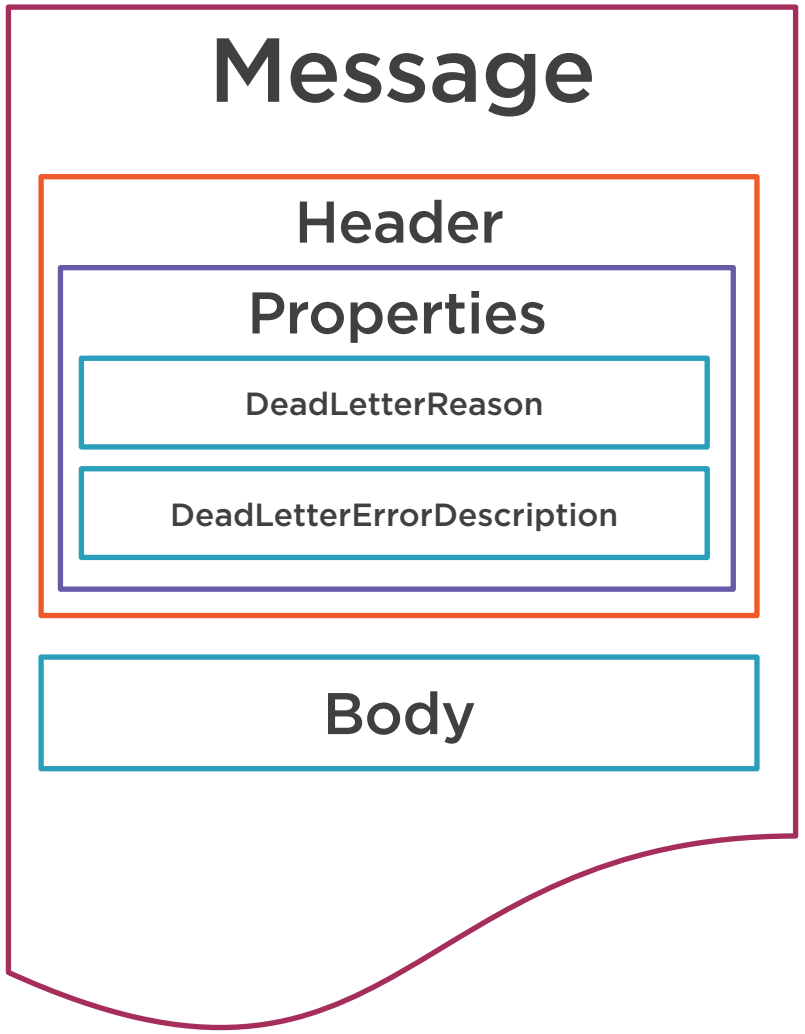


Explicitly Dead-lettering Messages

- **Call DeadLetter on received message**
  - Specify dead-letter reason and error description description

```
// Explicitly dead-letter a message.
await queueClient.DeadLetterAsync
    (message.SystemProperties.LockToken, "Invalid order", "Error in billing address");
```

Dead-Lettered Message Properties



- **Properties are added to the message properties collection**
  - **DeadLetterReason**
  - **DeadLetterErrorDescription**

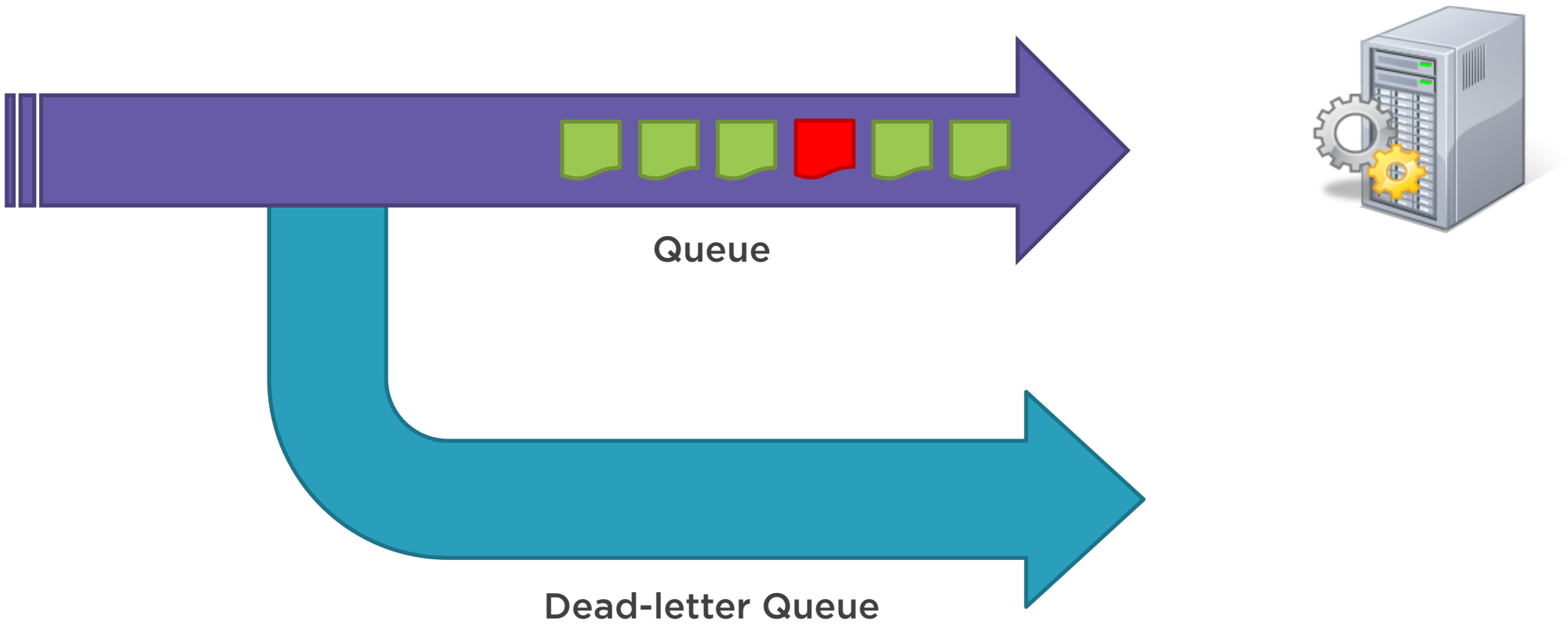
Demo

Demo: Dead-lettering Messages

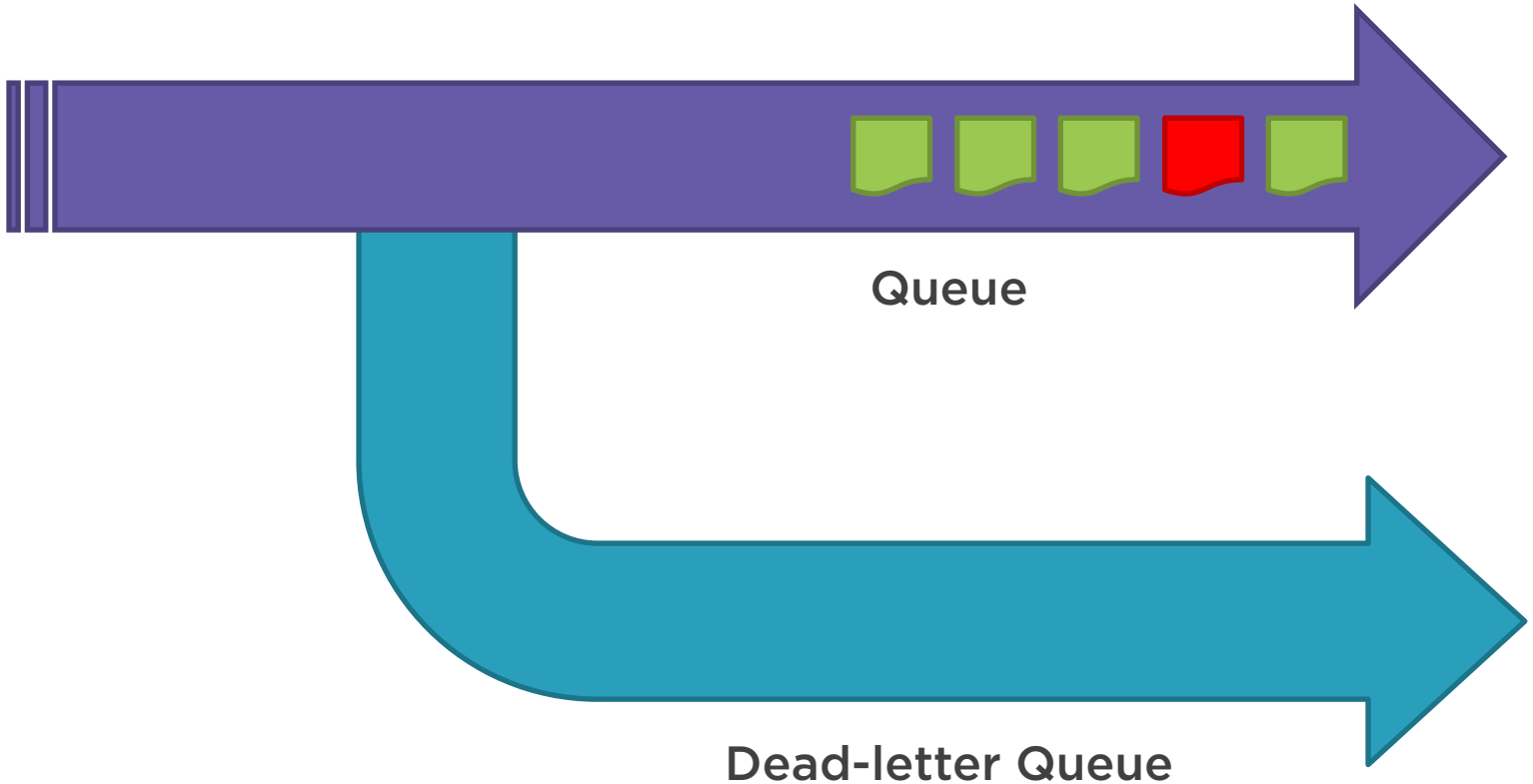
- Automatic Dead-lettering of Messages
- Message Delivery Count Property
- Explicit Dead-lettering of Messages

Handling Poison Messages

Poison Messages

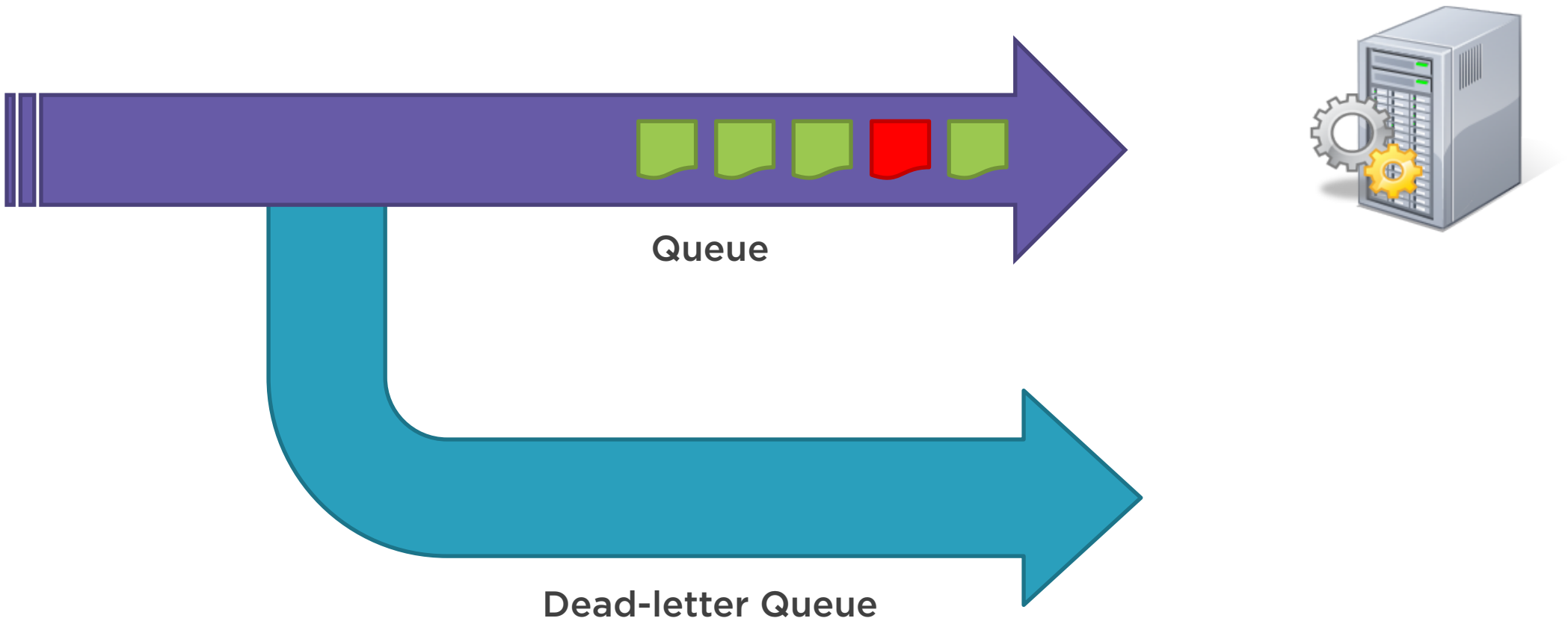


Poison Messages

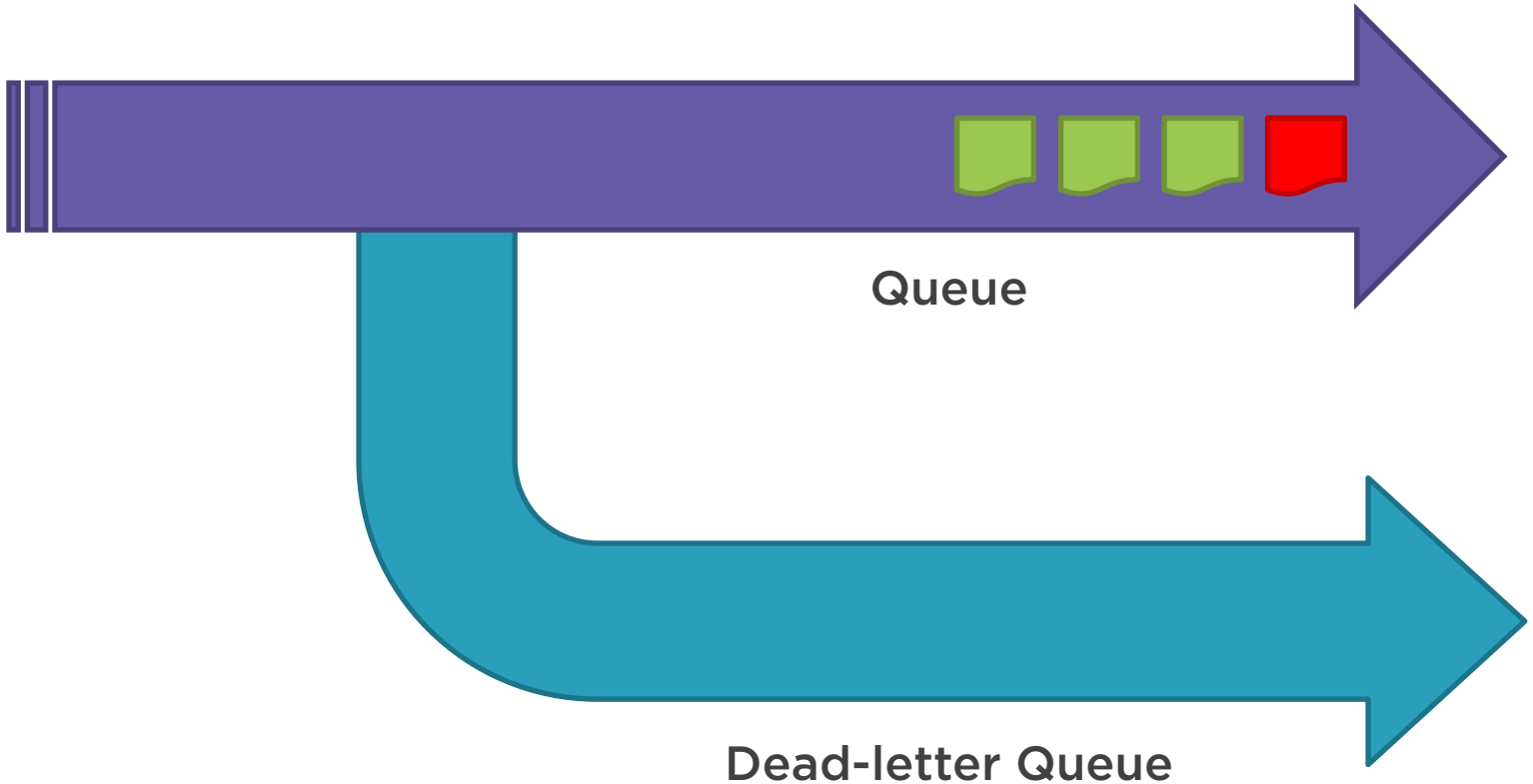


Process message

Poison Messages

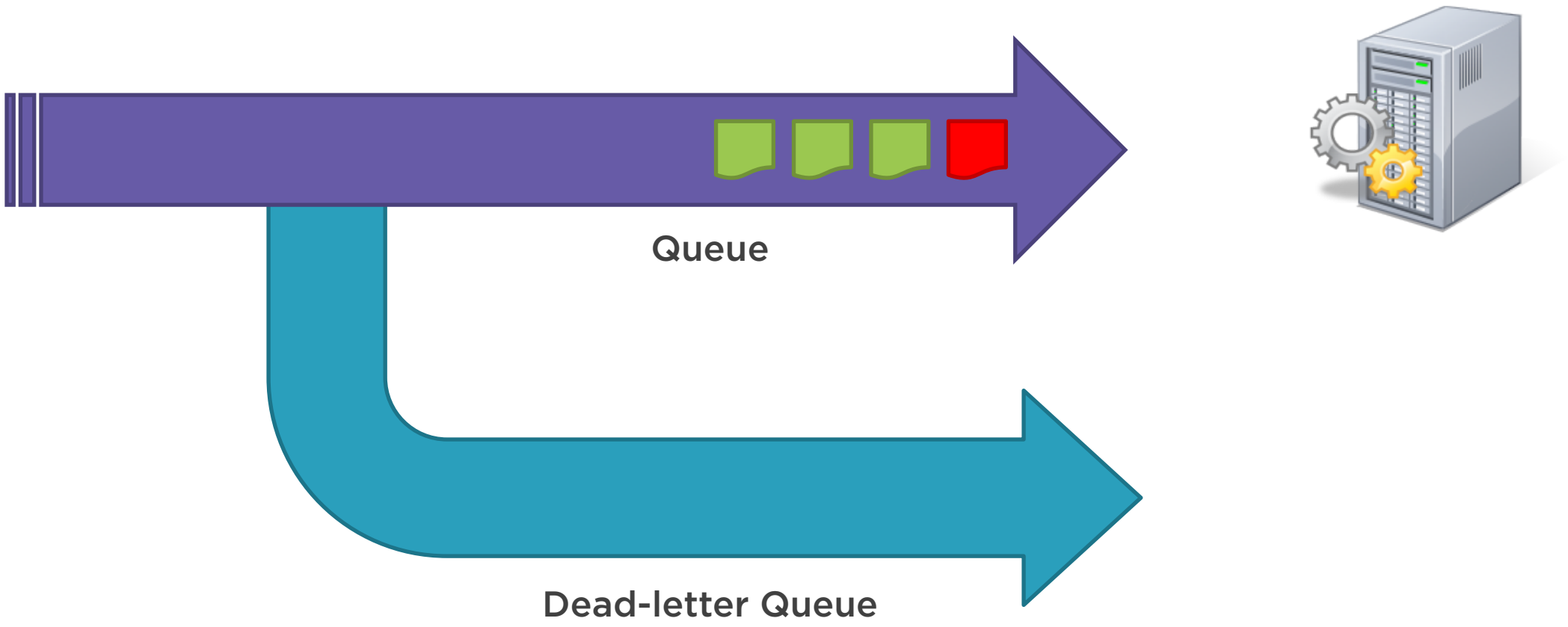


Poison Messages

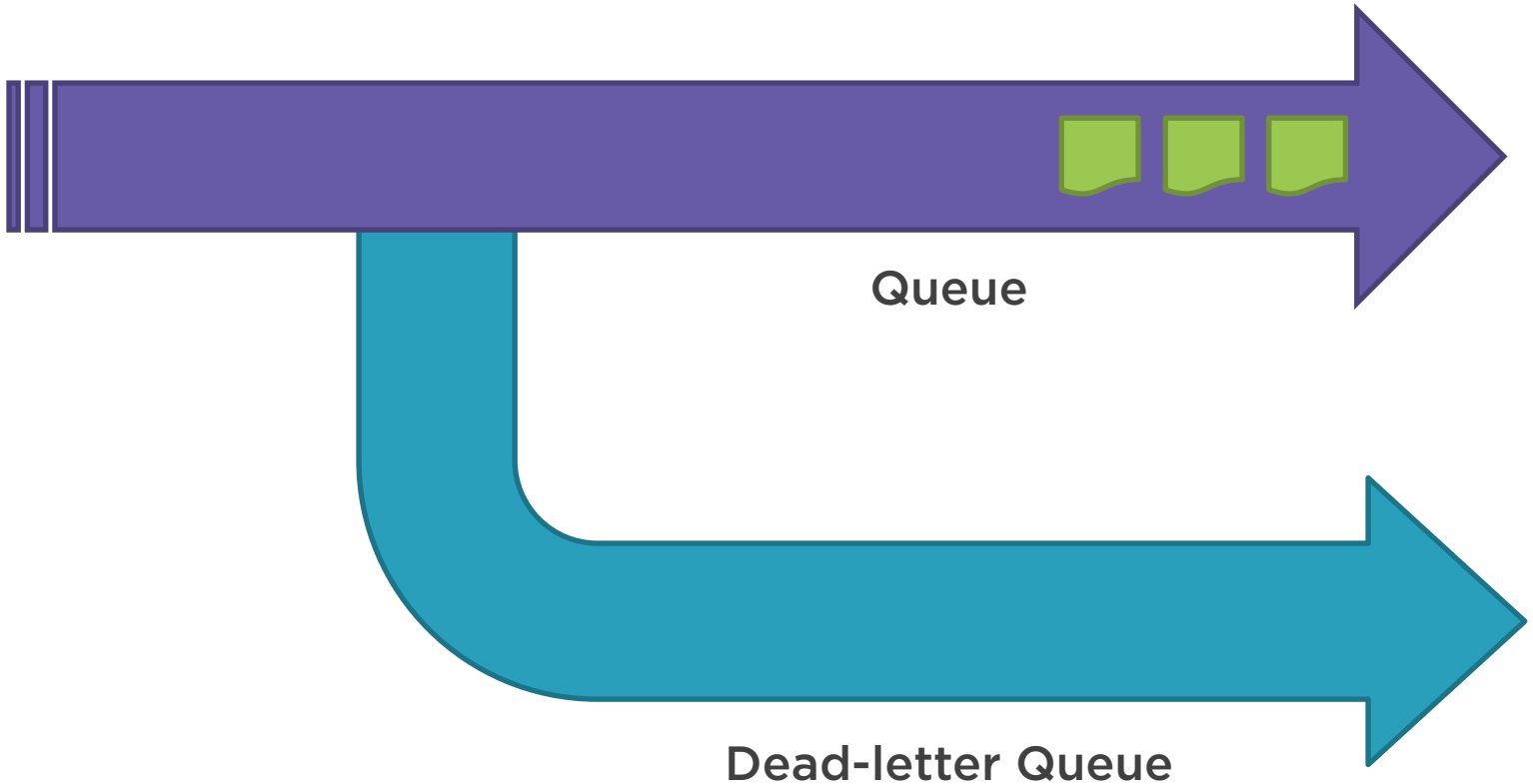


Process message

Poison Messages

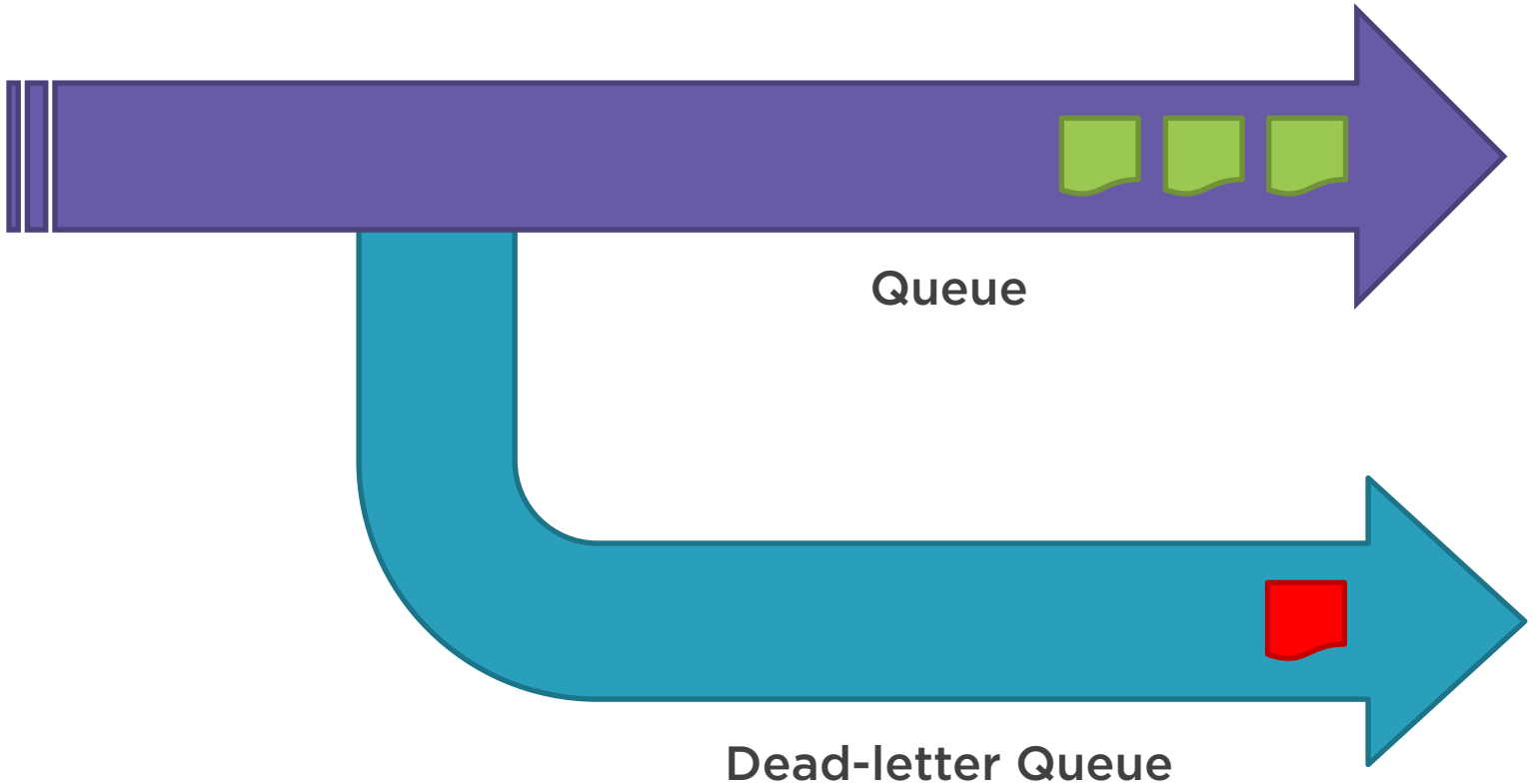


Poison Messages



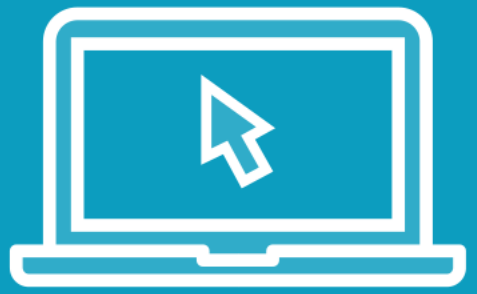
Error – Invalid message

Poison Messages



Dead-letter message

Demo



Handling Poison Messages

- Poison Messages
- Handling Poison Messages

Summary

Dead-Lettering Messages

- Don’t relay on max delivery count
- Take control of dead-lettering explicitly
- Providing reason and description of error
- Consider leveraging lock duration to handle transient faults

Handling Poison Messages

- Dead-letter explicitly providing reason and description of error