Practical MSMQ

Elton Stoneman
geekswithblogs.net/eltonstoneman
@EltonStoneman

pluralsight
hardcore dev and IT training

Practical MSMQ

Implement **request-response** & **publish-subscribe**

Interactive user requests & **event driven** workflows

Practical considerations for messaging

Request-response

Client builds and sends message
Specifies response queue address

Queue stores message

Handler reads message
Sends reply to response queue

Queue stores message

Client read response message
Processes reply

Request-Response

Request and response **message definitions**

Message handler query **logic** and **host**

New **queues** to isolate processing

Demo 1: Request-Response

Feature

Validate user's email address before unsubscribing

Task

Send request message from web & await response

Task

Process request message in handler & send response

Demo 1: Request-Response

Demo 1: Request-Response

- **Presentation layer**
  - Create dedicated response queue

```
var responseQueueAddress = string.Format(".\\private$\\{0}",
                                         Guid.NewGuid().ToString().Substring(0, 6));
var responseQueue = msmq.MessageQueue.Create(responseQueueAddress);
```

  - Send request message

```
using (var queue = new msmq.MessageQueue("...doesuserexist"))
{
   var message = new msmq.Message();
   message.BodyStream = doesUserExistRequest.ToJsonStream();
   message.Label = doesUserExistRequest.GetMessageType();
   message.ResponseQueue = responseQueue;
   queue.Send(message);
}
```

Demo 1: Request-Response

- **Presentation layer**
  - Listen on response queue

```
var response = responseQueue.Receive();
var doesUserExistResponse = response.BodyStream.ReadFromJson
                                                <DoesUserExistResponse>();
return doesUserExistResponse.Exists;
```

  - Delete response queue

```
if (msmq.MessageQueue.Exists(responseQueueAddress))
{
    msmq.MessageQueue.Delete(responseQueueAddress);
}
```

Demo 1: Request-Response

- **Message handler**
  - Receive & action request message

```
var message = queue.Receive();
var messageBody = message.BodyStream.ReadFromJson(message.Label);
//...
else if (messageBody.GetType() == typeof(DoesUserExistRequest))
{
    CheckUserExists((DoesUserExistRequest)messageBody, message);
```

  - Build & send response message

```
using (var queue = message.ResponseQueue)
{
    var response = new msmq.Message();
    response.BodyStream = responseBody.ToJsonStream();
    response.Label = responseBody.GetMessageType();
    queue.Send(response);
```

Considerations

Reliability

How long will the client wait?

Interactive may not need durability

Destination request queue

Shared queue for all clients

Split by priority

Destination response queue

Transient, one per request

Shared response queue, correlated

Publish-subscribe

Broadcast notifications
Publish events to users

Event-driven processing
Publish events to components

Decouple workflow stages
Parallel execution
Independent scaling
Extend workflow

Demo 2: Publish-Subscribe

Feature

Decouple unsubscribe workflow

Task

Existing handler to save changes & publish event

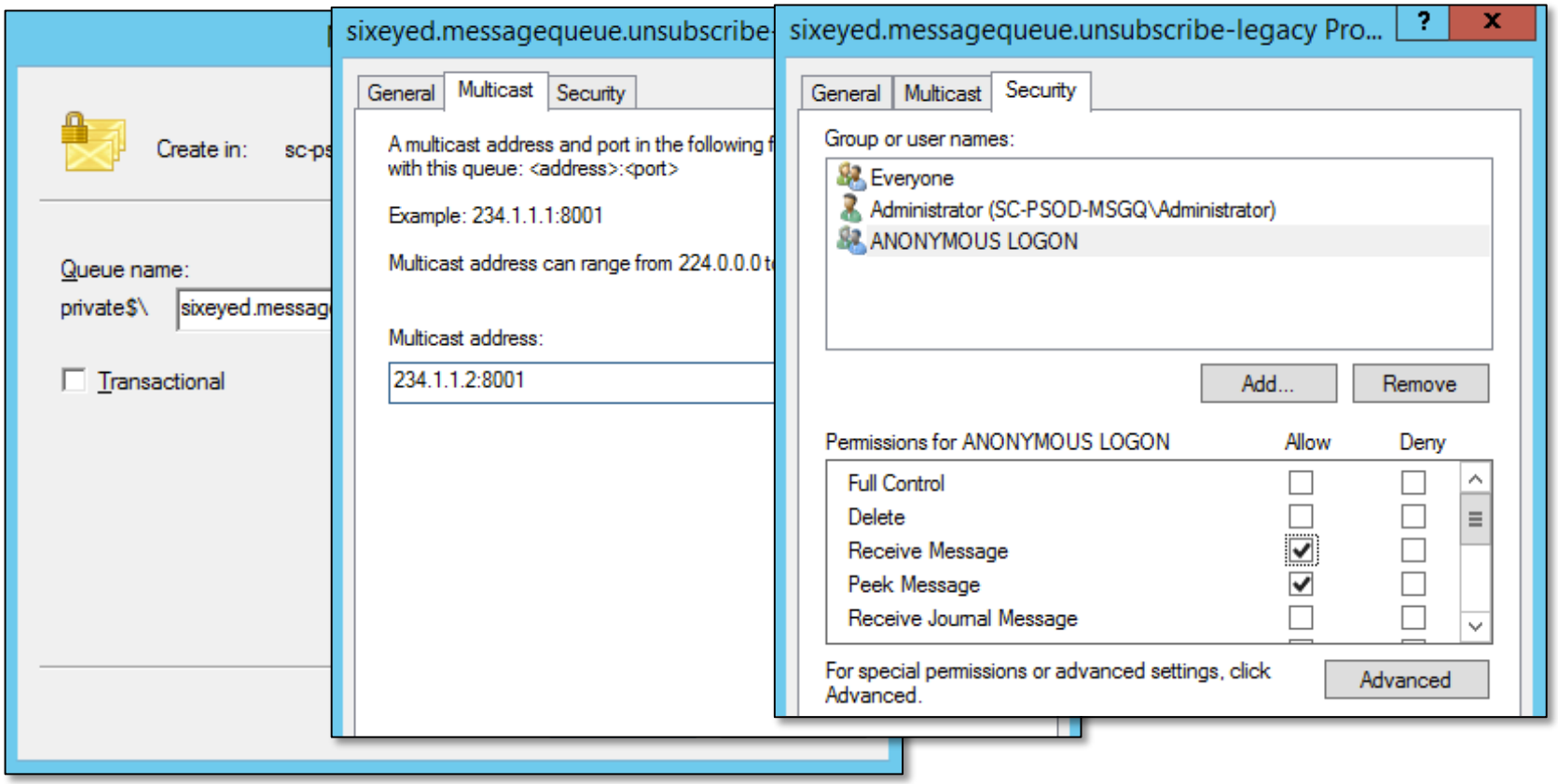
Task

New handlers subscribe to event & action workflow

Demo 2: Publish-Subscribe

Demo 2: Publish-Subscribe

- **Subscriber queue**
  - Non-transactional; multicast; ANONYMOUS LOGON



Demo 2: Publish-Subscribe

- **Unsubscribe workflow**
  - Persist unsubscribe status
  - Send user-unsubscribed event message

```
var unsubscribedEvent = new UserUnsubscribed
{
    EmailAddress = EmailAddress
};
using (var queue = new msmq.MessageQueue(
                                "FormatName:MULTICAST=234.1.1.2:8001"))
{
    var message = new msmq.Message();
    message.BodyStream = unsubscribedEvent.ToJsonStream();
    message.Label = unsubscribedEvent.GetMessageType();
    queue.Send(message);
}
```

Demo 2: Publish-Subscribe

- **Subscriber message handler**
  - Listen on multicast queue

```
var queueAddress = "...sixeyed.messagequeue.unsubscribe-legacy";
using (var queue = new msmq.MessageQueue(queueAddress))
{
    queue.MulticastAddress = "234.1.1.2:8001";
    while (true)
    {
        Console.WriteLine("Listening on: {0}", queueAddress);
        var message = queue.Receive();
```

Summary

- **Practical MSMQ** ☑
  - Implementing messaging patterns
- **Request-response** ☑
  - Interactive process
  - Dedicated request queue
  - Temporary response queue
- **Publish-subscribe** ☑
  - Decouple workflow components
  - Multicast over PGM
  - Separate queues & handlers

ZeroMQ