Introducing WebSphere MQ

Elton Stoneman
geekswithblogs.net/eltonstoneman
@EltonStoneman

pluralsight
hardcore dev and IT training

Introducing WebSphere MQ

Enterprise-grade messaging

Cross-platform server: Windows, Unix, Linux, IBM

Supports **complex configurations**

Goals

WebSphere MQ overview

Feature set and usage

Installation and configuration

Deployment on Windows

.NET client library

Structure and usage

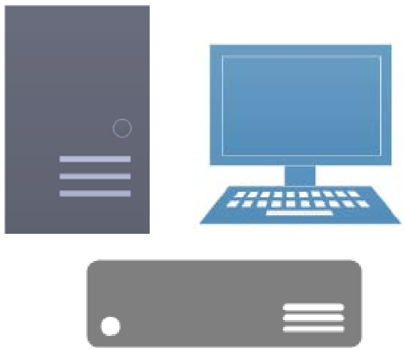
Pattern support

Fire-and-forget, request-response, publish-subscribe

What is WebSphere MQ?

p

On-premise
messaging

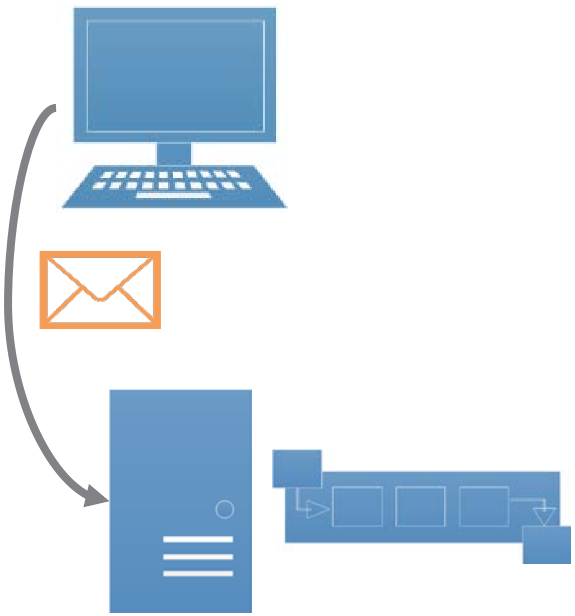


Client-server
platform support

3

Connected queue
technology

How Configurable is WebSphere MQ?

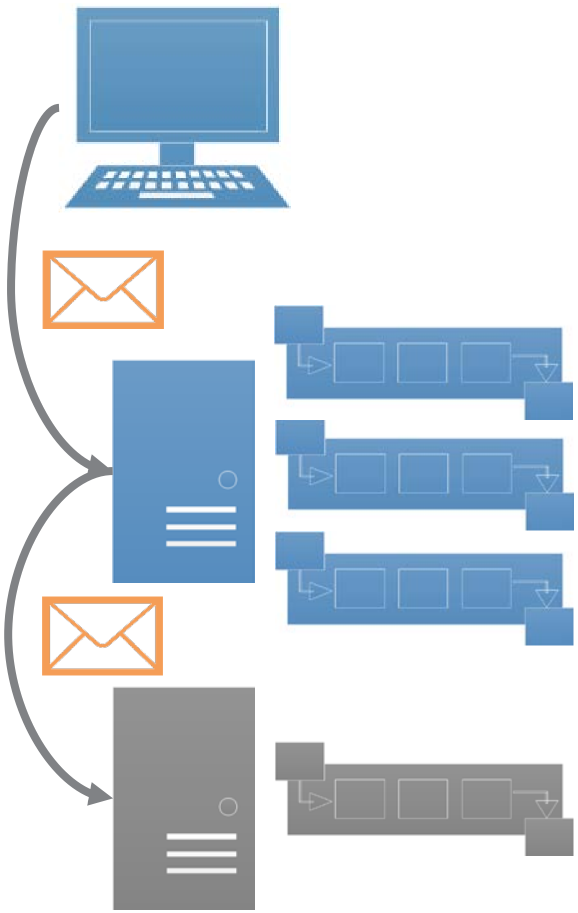


Queue Manager

Hosts queues

Manages client connections

How Configurable is WebSphere MQ?



Queue Manager

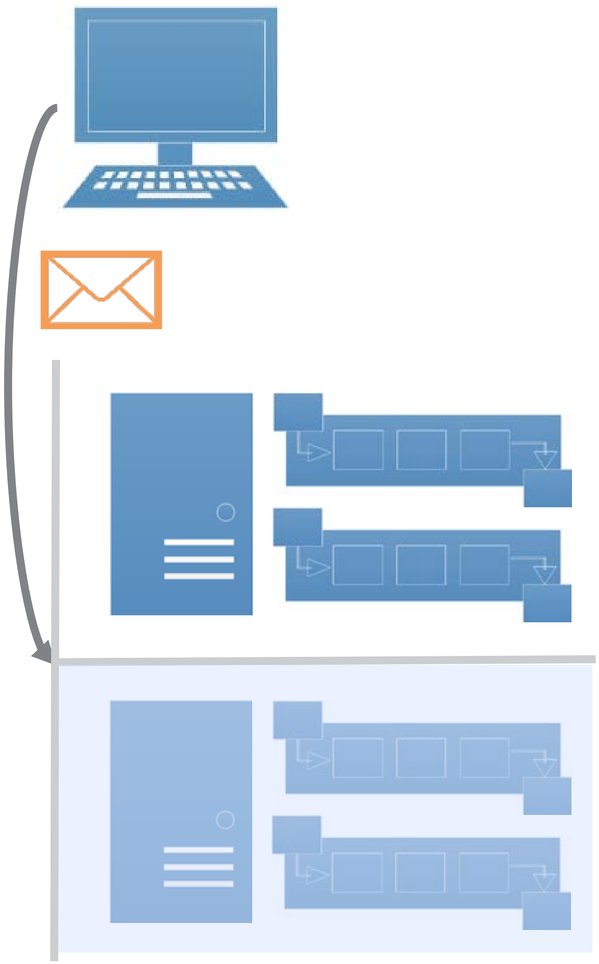
Hosts queues

Manages client connections

Remote queues

Remote Queue Manager

How Configurable is WebSphere MQ?



Queue Manager

Hosts queues

Manages client connections

Remote queues

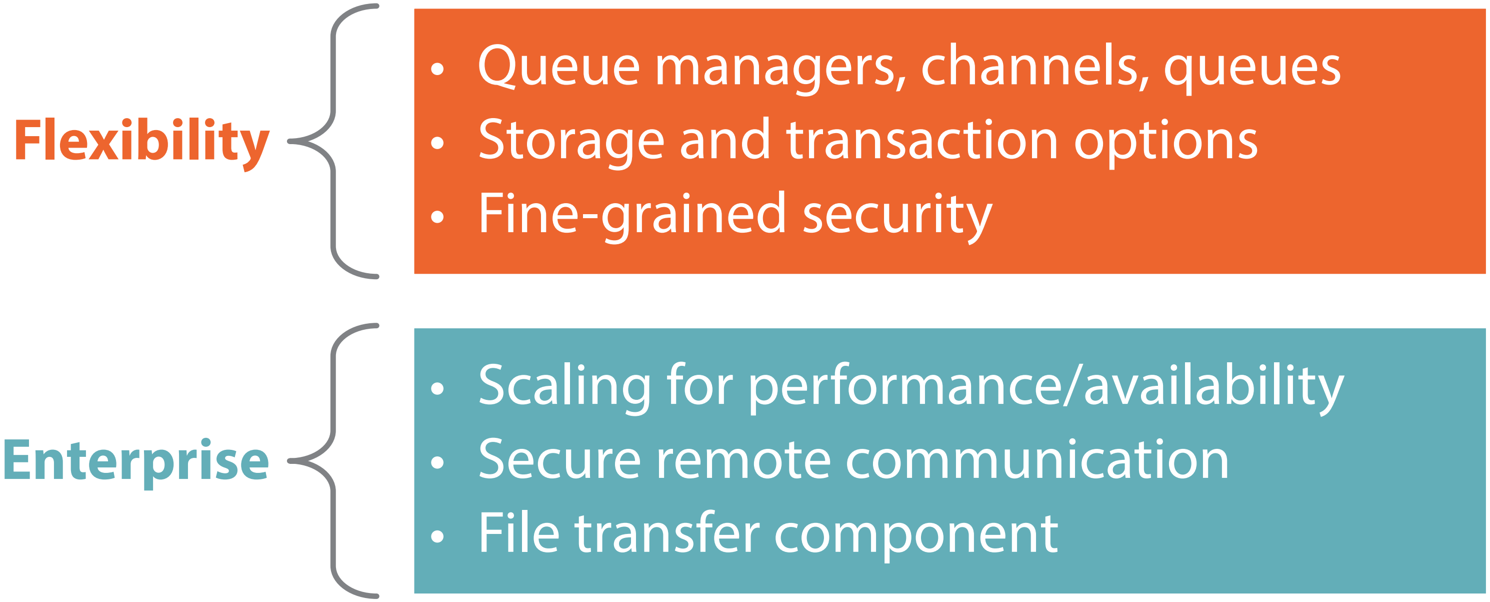
Remote Queue Manager

Clusters

High availability

Scale

WebSphere MQ Features



Flexibility

- Queue managers, channels, queues
- Storage and transaction options
- Fine-grained security

Enterprise

- Scaling for performance/availability
- Secure remote communication
- File transfer component

WebSphere MQ Features

Practical

- Dynamic response queues
- Optimized receive by correlation ID
- Large messages & segmentation

Administration

- Rich UI portal (Eclipse)
- Honours security model
- Command line (MQSC) & PowerShell

Demo 1: Deploying WebSphere MQ

Feature

Install and configure WebSphere MQ

Task

Install WebSphere MQ version 7.5 on Windows

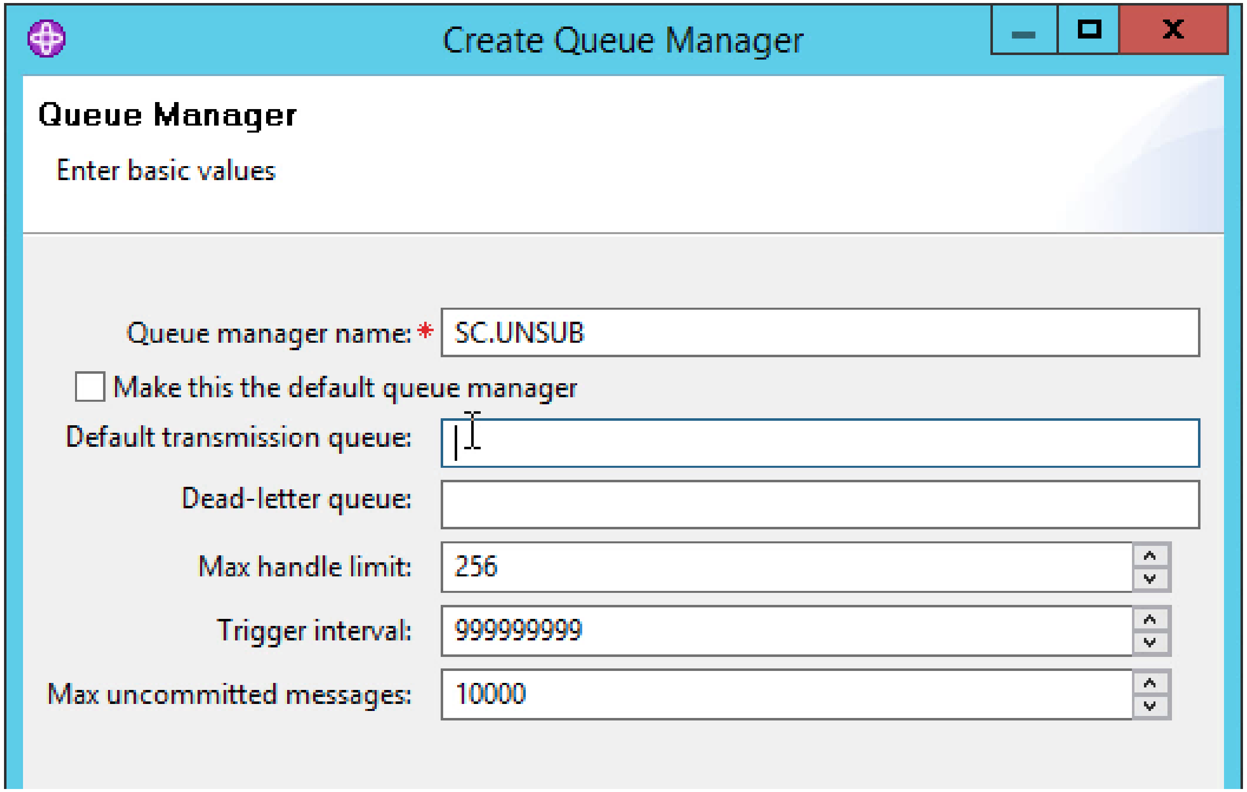
Task

Use MQ Explorer to create Queue Manager, Channel and Queue

Demo 1: Deploying WebSphere MQ

Demo 1: Deploying WebSphere MQ

- Create Queue Manager



Queue Manager

Enter basic values

Queue manager name: * SC.UNSUB

□ Make this the default queue manager

Default transmission queue:

Dead-letter queue:

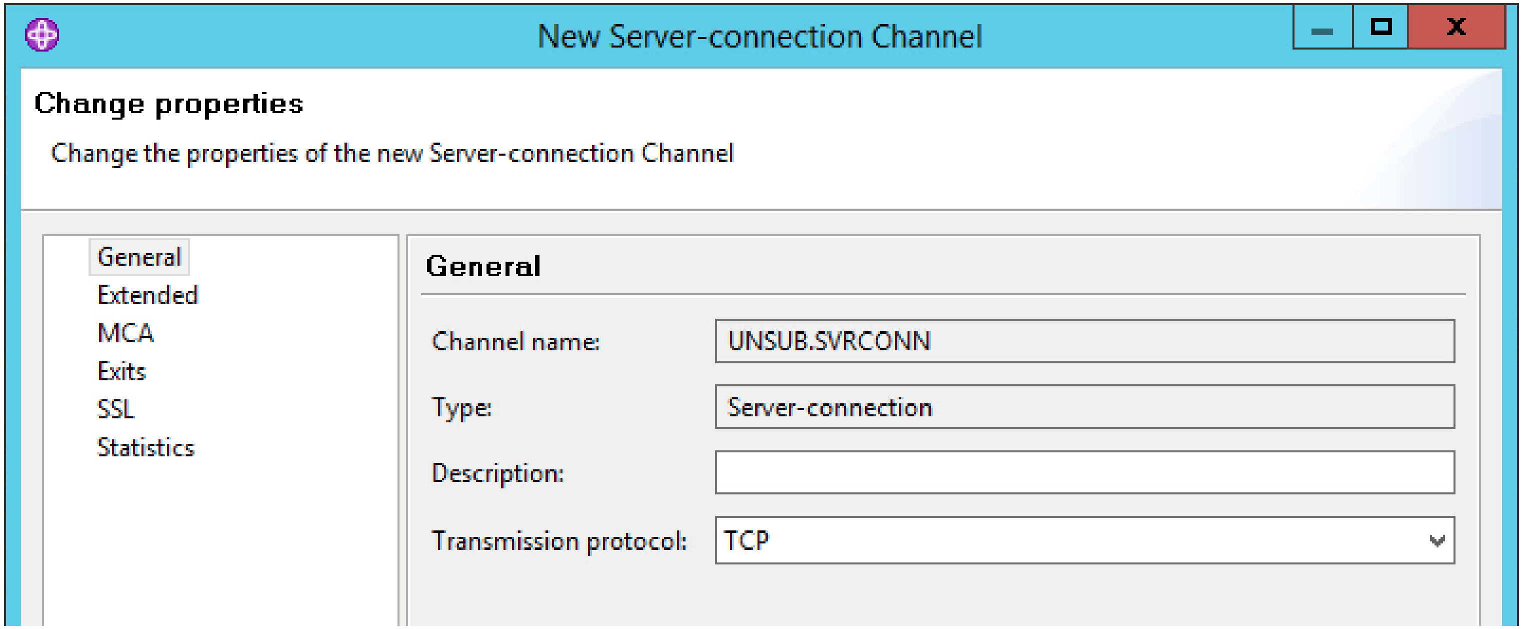
Max handle limit: 256

Trigger interval: 999999999

Max uncommitted messages: 10000

Demo 1: Deploying WebSphere MQ

- Channel Configuration



Change properties

Change the properties of the new Server-connection Channel

General

Channel name: UNSUB.SVRCONN

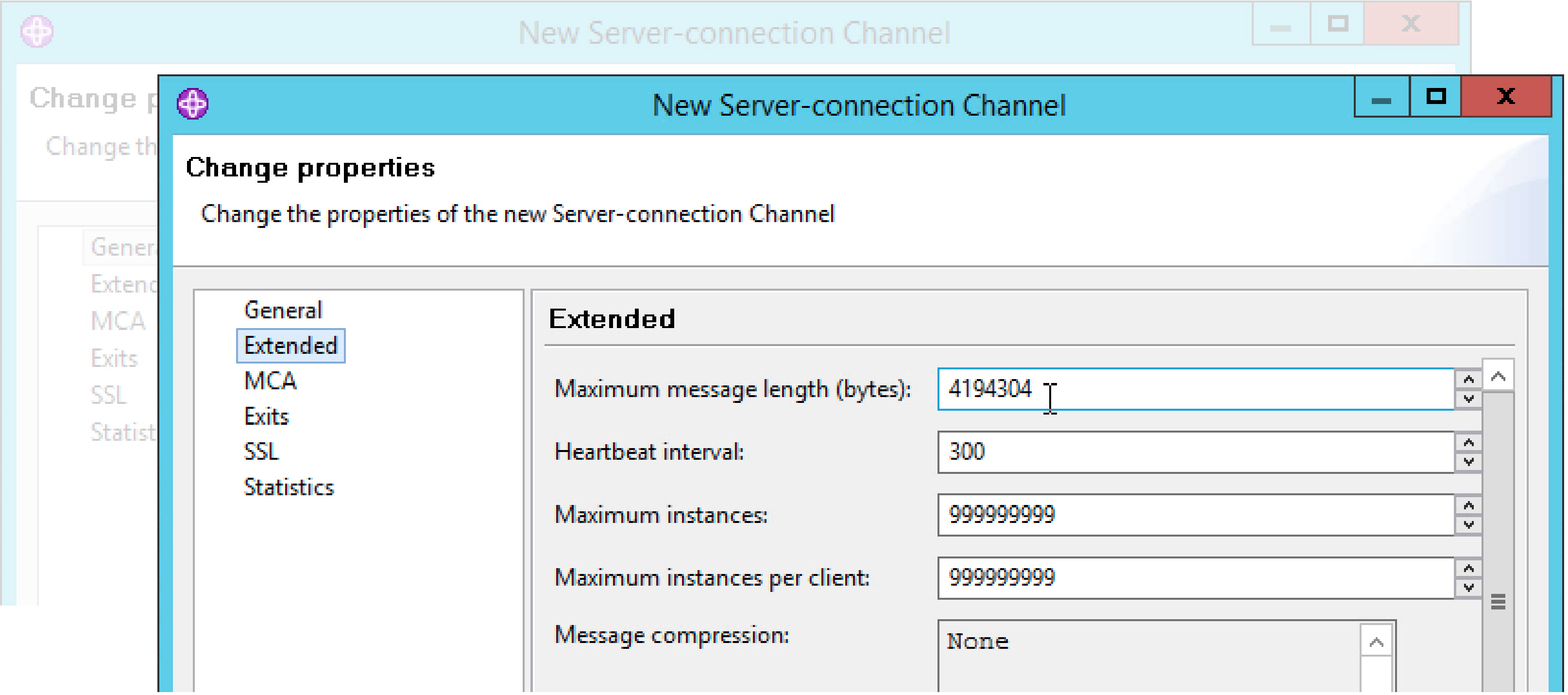
Type: Server-connection

Description:

Transmission protocol: TCP

Demo 1: Deploying WebSphere MQ

- **Channel Configuration**



New Server-connection Channel

Change properties

Change the properties of the new Server-connection Channel

Extended

Maximum message length (bytes): 4194304

Heartbeat interval: 300

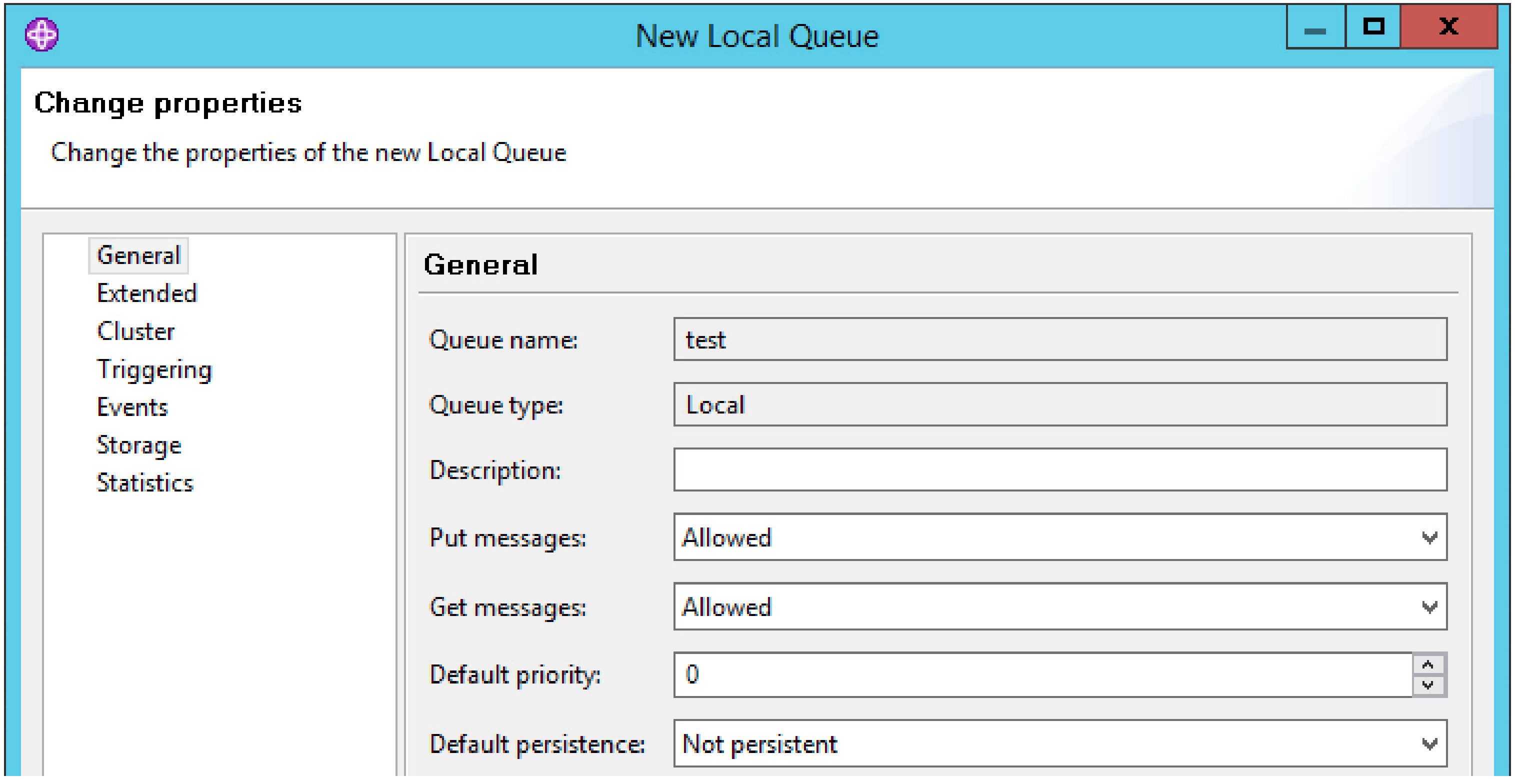
Maximum instances: 999999999

Maximum instances per client: 999999999

Message compression: None

Demo 1: Deploying WebSphere MQ

- Queue Configuration



Change properties

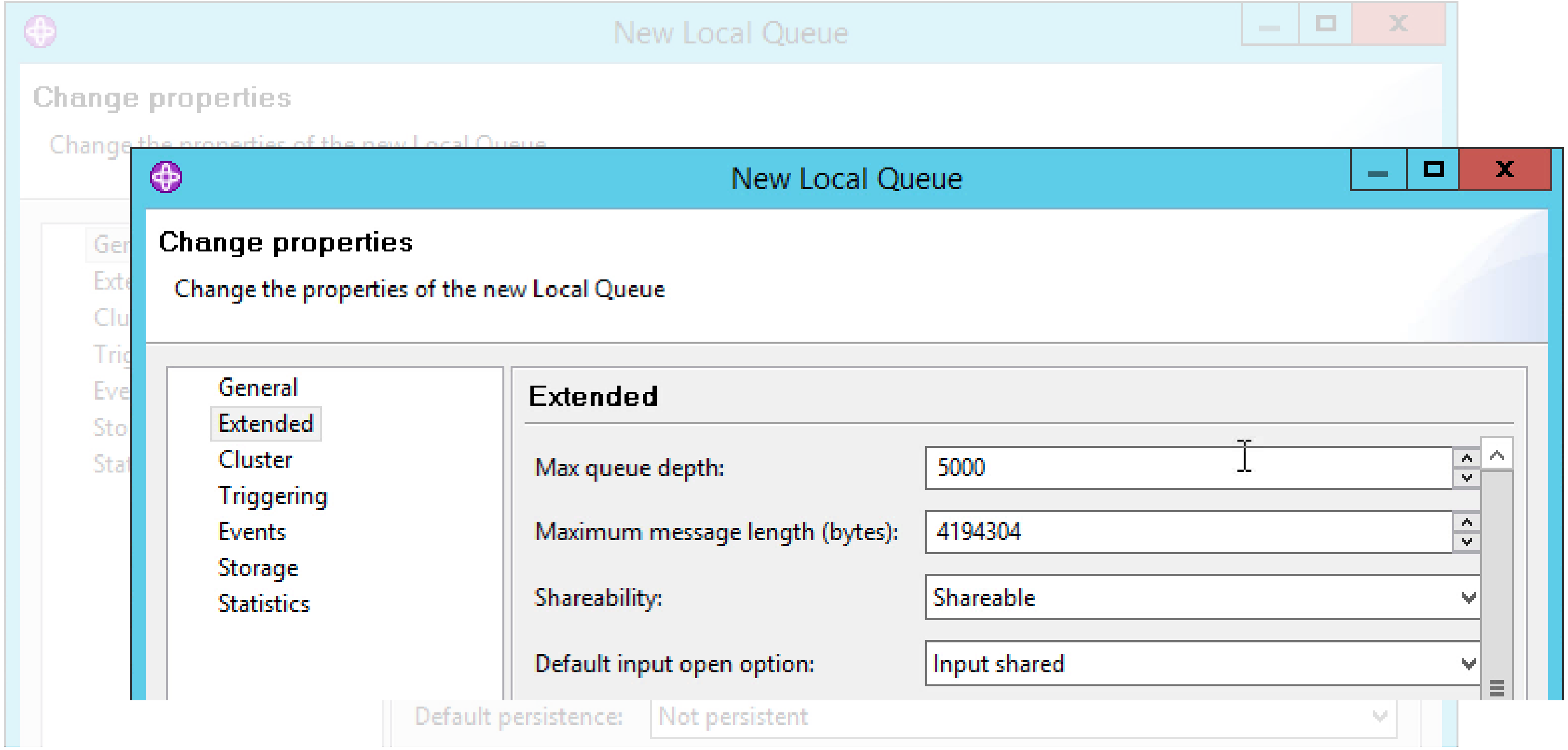
Change the properties of the new Local Queue

General

Queue name:	test
Queue type:	Local
Description:	
Put messages:	Allowed
Get messages:	Allowed
Default priority:	0
Default persistence:	Not persistent

Demo 1: Deploying WebSphere MQ

- **Queue Configuration**



New Local Queue

Change properties

Change the properties of the new Local Queue

Extended

Max queue depth: 5000

Maximum message length (bytes): 4194304

Shareability: Shareable

Default input open option: Input shared

Default persistence: Not persistent

Client Libraries

C

Client libraries for Java and .NET (etc.)

Java

JMS-style API for Java and .NET

IBM®

Native API - **feature parity** in .NET and Java

MQQueueManager

C

IBM.WMQ

Queue Manager connection

Specified channel

Connection properties

Administration

Change configuration

Create queues

Indirectly via model queues

MQQueue

C

IBM.WMQ

Messaging operations

Created from MQQueueManager

Context-bound to one queue

MQMessage

Define content format

Write body directly

Send and receive

Persistence options

Transactional options

Demo 2: Using WebSphere MQ with .NET

Feature

Explore messaging with WebSphere MQ and .NET

Task

Send and receive string messages with MQQueue

Task

Check basic performance for different reliability

Demo 2: Using WebSphere MQ with .NET

Demo 2: Using WebSphere MQ with .NET

- **Send messages**
  - Build connection properties for Queue Manager

```
var properties = new Hashtable();
properties.Add(MQC.HOST_NAME_PROPERTY, "127.0.0.1");
properties.Add(MQC.CHANNEL_PROPERTY, "UNSUB.SVRCONN");
```

  - Connect to Queue Manager & Queue
  - Specifying *MQOO_OUTPUT* option

```
var queueManager = new MQQueueManager("SC.UNSUB", properties);
var queue = queueManager.AccessQueue("test", MQC.MQOO_OUTPUT);
```

Demo 2: Using WebSphere MQ with .NET

- **Send messages**
  - □ Build message

```
var message = new MQMessage();
message.Format = MQC.MQFMT_STRING;
message.WriteString("message");
```

  - □ Send

```
queue.Put(message);
```

Demo 2: Using WebSphere MQ with .NET

- **Receive messages**
  - □ Build connection properties for Queue Manager

```
var properties = new Hashtable();
properties.Add(MQC.HOST_NAME_PROPERTY, "127.0.0.1");
properties.Add(MQC.CHANNEL_PROPERTY, "UNSUB.SVRCONN");
```

  - □ Connect to Queue Manager & Queue
  - □ Specifying *MQOO_INPUT** option

```
var queueManager = new MQQueueManager("SC.UNSUB", properties);
var queue = queueManager.AccessQueue("test", MQC.MQOO_INPUT_AS_Q_DEF);
```

Demo 2: Using WebSphere MQ with .NET

- **Receive messages**
  - Build message

```
var message = new MQMessage();
message.Format = MQC.MQFMT_STRING;
message.WriteString("message");
```

  - Receive

```
queue.Get(message);
```

  - Read body

```
message.ReadString(message.MessageLength).Dump("Body");
```

Demo 2: Using WebSphere MQ with .NET

- **Transactional messaging**
  - Specify *SYNCPOINT* option on Put

```
var options = new MQPutMessageOptions()
{
        Options = MQC.MQPMO_SYNCPOINT
};
queue.Put(message, options);
```

  - Commit

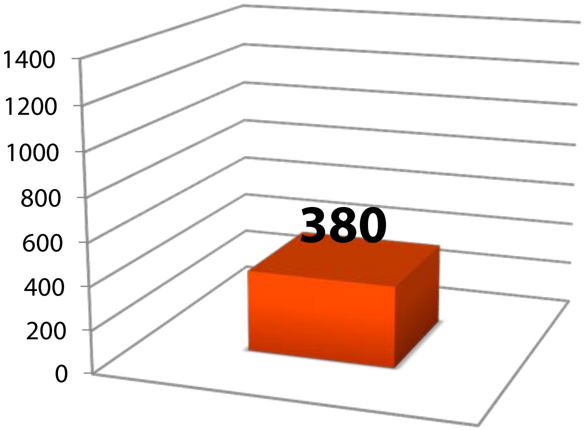
```
queueManager.Commit();
```

Demo 2: Using WebSphere MQ with .NET

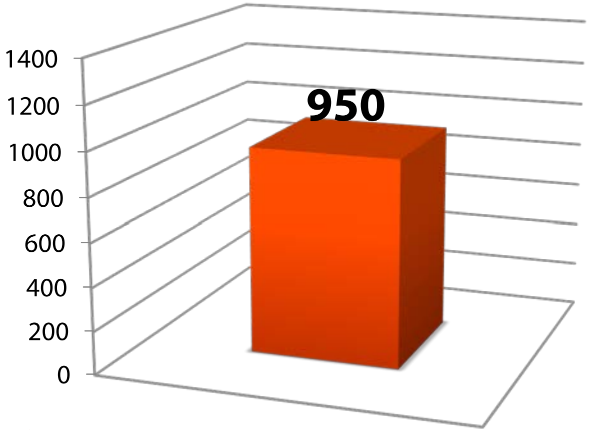
- **Durable messaging**
  - □ Specify Persistence option on message

```
var message = new MQMessage();
message.Persistence = MQC.MQPER_NOT_PERSISTENT; //0;
message.Persistence = MQC.MQPER_PERSISTENT; //1
message.Persistence = MQC.MQPER_PERSISTENCE_AS_Q_DEF; //2;
```

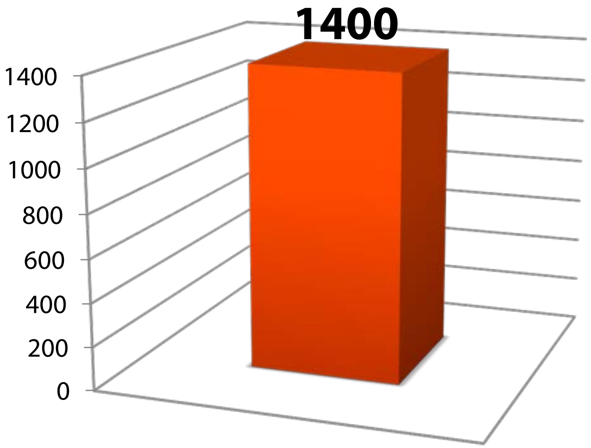
Demo 2: Using WebSphere MQ with .NET



Queue default
non-persistent



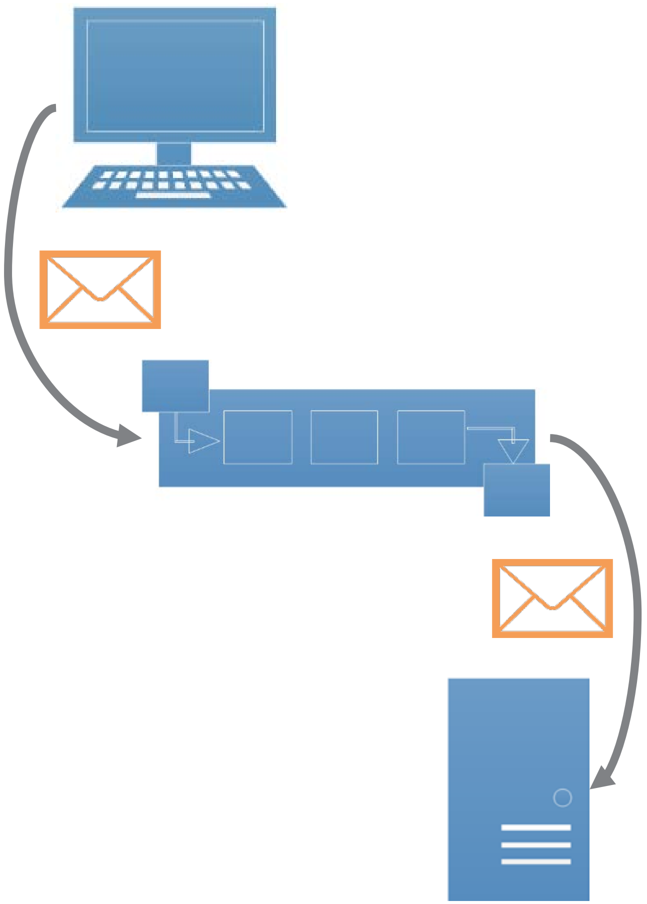
Persistent
messages



Syncpoint
queue access

🟧 - milliseconds to send 1,000 messages

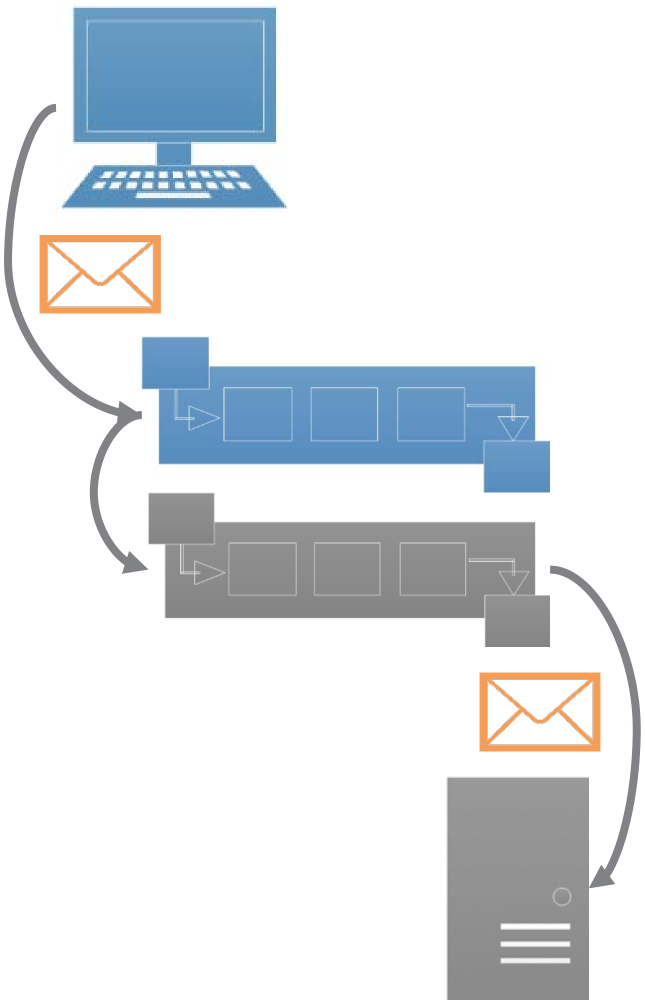
Messaging Pattern Support



Fire-and-forget

Standard support

Messaging Pattern Support

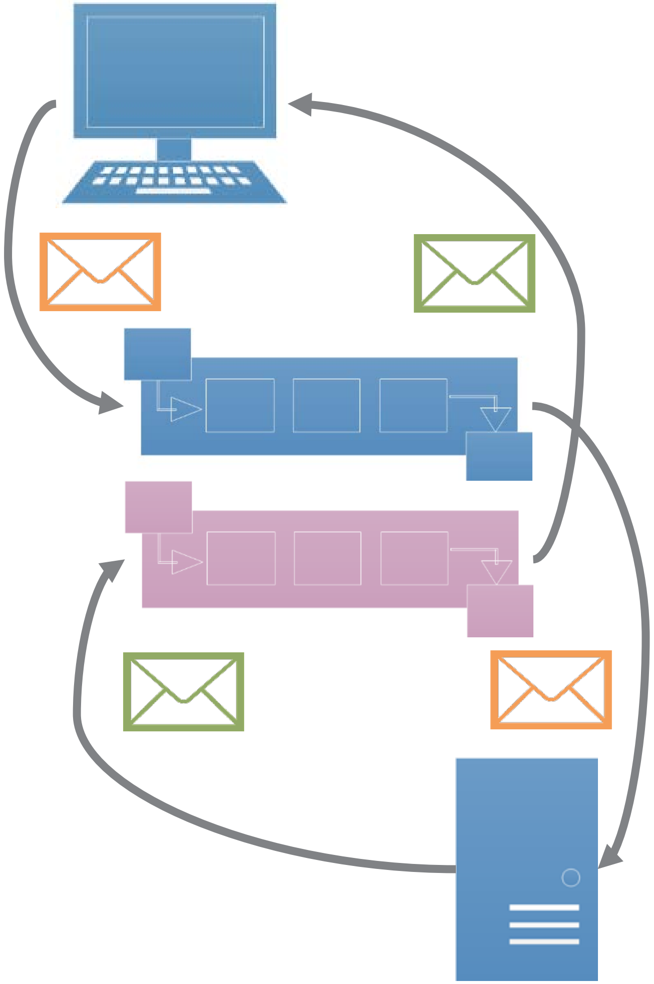


Fire-and-forget

Standard support

B2B with remote queues

Messaging Pattern Support



Fire-and-forget

Standard support

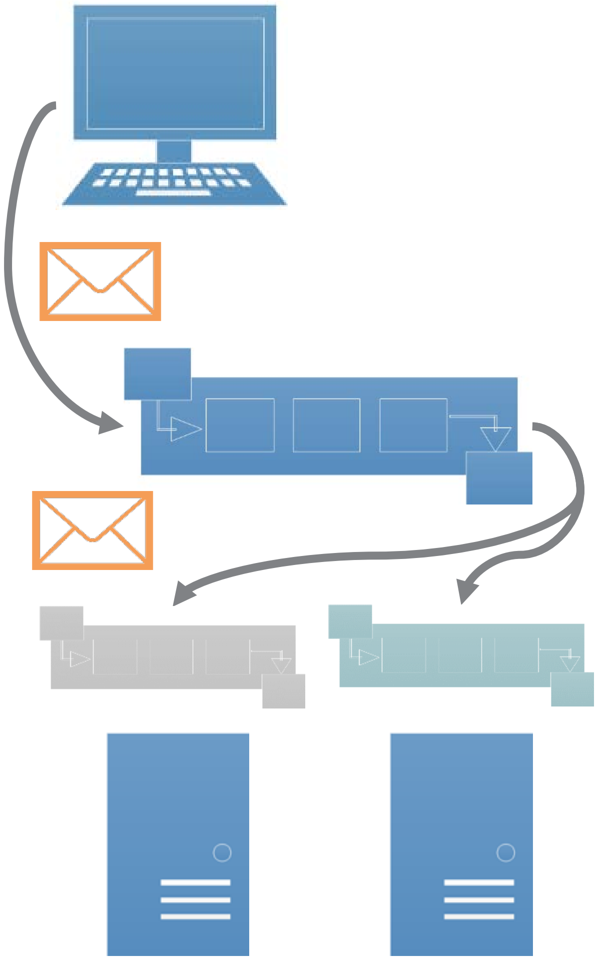
B2B with remote queues

Request-response

Shared response queue

Dynamic temporary response queue

Messaging Pattern Support



Fire-and-forget

Standard support

B2B with remote queues

Request-response

Shared response queue

Dynamic temporary response queue

Publish-subscribe

Topics coupled to queues

Summary

- **Introducing WebSphere MQ** ☑
  - Enterprise-grade messaging
  - Cross-platform client-server
- **Components** ☑
  - Queue Manager & channel
  - Queue – local, alias, model, remote
  - And lots more options
- **Usage** ☑
  - MQ Explorer
  - .NET client library
  - Performance

WebSphere MQ