Hybrid messaging with MQ Light, MQ's beta support for AMQP, and Bluemix

Matthew Whitehead
WebSphere MQ Development
mwhitehead@uk.ibm.com
Agenda

• Quick recap of MQ Light

• How MQ Light and AMQP applications work with IBM MQ (beta)

• Hybrid Messaging with MQ and Bluemix
IBM Messaging

Deliver Messaging Backbone for Enterprise

Focus on traditional MQ values, rock-solid enterprise-class service, ease-of-operation, breadth of platform coverage, availability, z/OS exploitation

Enable Developers to build more scalable, responsive applications

Focus on app use cases, breadth of languages, ease-of-deployment, micro services, integration with developer frameworks
MQ Light : Software and Cloud

- Messaging that application developers will love to use, helping them make responsive applications that scale easily

- 3 ways to get it:
  - Bluemix service
  - MQ Light software download
  - MQ Version 8.0.0.3 beta support for MQ Light

- Open APIs crafted to feel natural in a growing range of popular languages

- Tooling that makes modular app development easy

THIS PRESENTATION
The journey that got us here

**I want to execute code without taxing my Web app processes**

[Illustration of a flowchart showing My Apps, Some Thing, and Workers]

**Andy**
Developer

**Iain**
Infrastructure Guy

**My job is run a communications service for my customers’ apps**

[Illustration showing Messaging Backbone and connections to My Customers’ Apps]
Use cases

**Worker offload**
- Image processing
- Text analytics

**Event driven**
- Posting video to multiple social sites after transcoding
- Respond to external events

**Connecting external systems**
- Updating external booking app
- Posting updates to Twitter
Deployment Options

“MQ Light”

WebSphere MQ
[open Beta]

MQ Light Service for Bluemix
MQ Light API - Language support

- Simple, programming Language neutral **messaging model**

- Idiomatic language & framework **API Mappings**
  - Frictionless development

- Open wire protocol
  - Open Source client libraries
  - Facilitates community drivers for languages & frameworks
MQ Light Messaging Model

Sender application

1. Send ('/test/a', "Hello");
2. Send ('/test/a', "World!");

Topic Address Space

DESTINATION
Pattern=/test/#

SHARING
Share=myshare

Client 1
1. Hello
2. World!

Client 2
1. Hello

Client 3
2. World!
Agenda

• Quick recap of MQ Light

• How MQ Light and AMQP applications work with IBM MQ (beta)

• Hybrid Messaging with MQ and Bluemix
MQ Light Support in IBM MQ

- MQ V8 Announce
- Statement of Direction - MQ Light Support in IBM MQ
- MQ Light Standalone Beta

MQ V8.0.0.2
- Release: MQ AMQP Tech Preview
- Sept 2014

MQ V8.0.0.3
- Update: MQ AMQP Tech Preview
- Feb 2015

IBM MQ V8.0.0.3
- June 2015

IBM MQ V8.0.0.2
- Sept 2014

MQ Light Standalone GA
- MQ Light Bluemix Service GA
- April 2014

MQ Light Standalone Beta
- MQ Light Standalone Alpha
- Feb 2014

MQ Light Standalone Alpha
- Feb 2014

MQ Light Standalone Beta
- MQ Light Standalone GA
- Sept 2014
IBM MQ – MQ Light Tech Preview

- **Platforms**
  - Windows 64 Bit
  - Linux x86_64

- **Beta Installation**
  - Prereq is IBM MQ V 8.0.0.3
  - Add Tech Preview install media
    - Linux – RPM which is installed along side the other MQ RPMs
    - Windows – Zip which is manually extracted to an MQ installation
New AMQP channel type

- Adds a channel type of “AMQP”
- Support a subset of the AMQP 1.0 Oasis specification
- Interoperable with MQ FAP and MQTT applications (see later slides for details)
The architecture of an MQ server-conn channel

MQ App

MQ Client

MQ Listener

TCP Port

MQ SVRCONN Channel

HCONN

QM

0..n

1

0..n

1

1

0..n

1

1

0..n

1

1

0..n

1
The architecture of an MQ Light channel

**MQ Light Client**
- 1
- 0..n
- Client ID
- 1
- 1
- HCONN
- 0..n
- (For consuming messages)
- 1
- QM
- 1
- 1
- HCONN
- 0..n
- 1
- 1
- AMQP Channel
- 1
- TCP Port
- 1
- 1
- HCONN Pool
- 4
- (For publishing messages)
- 1
- JVM Process
- 1
- MQ Service
- 1
- 1
- (Beta)
AMQP channels

- **Configuration model**
  - MQSC and PCF updates allow you to administer AMQP channels in much the same way as other MQ objects
    
    - DISPLAY CHANNEL(*) CHLTYPE(AMQP)
    - DEFINE CHANNEL(MY.AMQP.CHANNEL) CHLTYPE(AMQP) PORT(5673)
    - START CHANNEL(MY.AMQP.CHANNEL)
    - STOP CHANNEL(MY.AMQP.CHANNEL)
    - DISPLAY CHSTATUS(*) CHLTYPE(AMQP)

- PCF command types all valid, using MQIACH_CHANNEL_TYPE=MQCHT_AMQP:
  
  - MQCMD_CREATE_CHANNEL
  - MQCMD_CHANGE_CHANNEL
  - MQCMD_STOP_CHANNEL
  - MQCMD_INQUIRE_CHANNEL
  - MQCMD_INQUIRE_CHANNEL_STATUS
  
  - MQCMD_DELETE_CHANNEL
  - MQCMD_START_CHANNEL
  - MQCMD_COPY_CHANNEL
AMQP channels

Note the new client ID attribute set on the MQ connection
Note, DIS CHSTATUS usage varies slightly from MQ channels
AMQP channels displayed alongside existing channels

<table>
<thead>
<tr>
<th>Channel name</th>
<th>Channel type</th>
<th>Overall status</th>
</tr>
</thead>
<tbody>
<tr>
<td>MYAMQPCHANNEL</td>
<td>AMQP</td>
<td>Running</td>
</tr>
<tr>
<td>SYSTEM.AUTO.RECEIVER</td>
<td>Receiver</td>
<td>Inactive</td>
</tr>
<tr>
<td>SYSTEM.AUTO.SVRCONN</td>
<td>Server-connection</td>
<td>Inactive</td>
</tr>
<tr>
<td>SYSTEM.DEF.AMQP</td>
<td>AMQP</td>
<td>Running</td>
</tr>
<tr>
<td>SYSTEM.DEF.CLUSRCVR</td>
<td>Cluster-receiver</td>
<td>Inactive</td>
</tr>
<tr>
<td>SYSTEM.DEF.CLUSSDR</td>
<td>Cluster-sender</td>
<td>Inactive</td>
</tr>
<tr>
<td>SYSTEM.DEF.RECEIVER</td>
<td>Receiver</td>
<td>Inactive</td>
</tr>
<tr>
<td>SYSTEM.DEF.REQUESTER</td>
<td>Requester</td>
<td>Inactive</td>
</tr>
<tr>
<td>SYSTEM.DEF.SENDER</td>
<td>Sender</td>
<td>Inactive</td>
</tr>
<tr>
<td>SYSTEM.DEF.SERVER</td>
<td>Server</td>
<td>Inactive</td>
</tr>
<tr>
<td>SYSTEM.DEF.SVRCONN</td>
<td>Server-connection</td>
<td>Inactive</td>
</tr>
</tbody>
</table>
Application connections view use to display AMQP clients

- Connection name and channel name populated to show where the client has connected from
- New **Client ID** attribute in the Application Connections view
Authorities and authentication

- MQ authenticates the client and authorises messaging in a similar way to MQ clients. Similarly, administration changes and commands for AMQP channels are authorised in the same way as for other MQ channels. Allows you to specify who has authority to:

  - Start or stop a channel
  - Change a channel's configuration
  - Display a channel's status
  - Delete a channel

Client opens SSL/TLS connection to MQ. Certs checked.

MQ channel auth rules checked (IP address valid, SSL/TLS certificate DN valid etc.)

AMQP SASL frame arrives at the queue manager. MQ conn auth configuration checked.

MQ 'connect', 'publish', 'subscribe' authorities checked for resulting MQ user ID.

Channel MCAUSER asserted, unless channel auth rules determine otherwise.

JAAS module invoked.

MQ security exit definitions invoked.

Connection established with the assumed MQ identity.
Monitoring

• Events

  • MQ provides events for monitoring different activities

  • Some are available to try in the beta
    - Command events (e.g. request to start a channel)
    - Configuration events (e.g. request to change channel attrs)

  • Some are ones we'd like to do
    - Security events (e.g. an AMQP client failed an authority check)
Backup/Restore

- MQ provides tools to saving and restoring queue manager configuration
  - dmpmqcfg and runmqsc
  - These have been updated to include AMQP channel definitions
  - Service may request that custom tuning/service parameters be set in a .properties file. If so, then this file must be manually backed up similar to qm.ini files today

Logs

- Located in MQ data path
  - /var/mqm/qmgrs/QM1/amqp_stdout and /var/mqm/qmgrs/QM1/amqp_stderr
  - /var/mqm/qmgrs/errors/amqp_*\.log
  - /var/mqm/trace/amqp_*\.trc (start/end trace using strmqtrc/endmqtrc)
Interoperability

- AMQP to MQ FAP and MQTT

**Published** AMQP messages MQPUT to an MQ topic

**Consumed** AMQP messages MQGET from an MQSUB

MQ apps can **consume** AMQP publishes by subscribing to matching topic pattern

MQ apps can **publish** to AMQP clients by MQPUT to the same topic string

warehouse/item/372837  warehouse/item/#

orders/electrical/#  orders/electrical/wiring
Interoperability

- AMQP publisher to MQ consumer
  - MQMD PutApplType always set to MQAT_AMQP
  - Some AMQP attributes → MQMD
  - Some AMQP attributes → MQ message properties
  - All AMQP application properties → MQ message properties
  - Simple AMQP binary payload → MQFMT_NONE message
  - Simple AMQP string payload → MQFMT_STRING message
  - All other AMQP payloads → MQFMT_AMQP
Interoperability

- MQ publisher to AMQP consumer
  - Some MQMD fields → AMQP headers
  - Some MQMD fields → AMQP properties
  - All MQ message properties → AMQP application properties
  - MQFMT_NONE message → single AMQP binary data payload
  - MQFMT_STRING message → single AMQP string data payload
  - MQFMT_AMQP message → copy to payload section

Note: All MQ messages are got with MQGMO_CONVERT to convert string data to UTF8
Some AMQP **headers** are set as MQMD fields:

- AMQP header.ttl set on MQ message as MQMD.expiry (converted to 10ths of a second)
- AMQP header.priority set on MQ message as MQMD.priority (max value of 9)
- AMQP properties.correlation-id set on MQ message as MQMD.correlid

All AMQP **headers** are set as MQ message properties with a mapped name:

- header.durable set on MQ message as MQ property AMQPDurable
- header.priority set on MQ message as MQ property AMQPPriority
- header.ttl set on MQ message as MQ property AMQPTtl
- header.first-acquirer set on MQ message as MQ property AMQPFirstAcquirer
- header.delivery-count set on MQ message as MQ property AMQPDeliveryCount

All AMQP **properties** are also set as MQ message properties, e.g.

- properties.user-id set on MQ message as MQ property AMQPUserId
- properties.to set on MQ message as MQ property AMQPTo
- properties.subject set on MQ message as MQ property AMQPSubject
- properties.reply-to set on MQ message as MQ property AMQPReplyTo
- properties.content-type set on MQ message as MQ property AMQPContentType
- properties.content-encoding set on MQ message as MQ property AMQPContentEncoding
- properties.creation-time set on MQ message as MQ property AMQPCreationTime
- properties.group-id set on MQ message as MQ property AMQPGroupId
- properties.message-id set on MQ message as MQ property AMQPMessageId
- Properties.group-sequence set on MQ message as MQ property AMQPGroupSequence
- Properties.absolute-expiry-time set on MQ message as MQ property AMQPAbsoluteExpiryTime
- Properties.reply-to-group-id set on MQ message as MQ property AMQPReplyToGroupId

Finally, all AMQP **application-properties** are copied into the MQ message properties in the user space (usr.*) using similar naming conventions with some restrictions on the property length, characters used, and certain keyword restrictions e.g. “JMS”.

*(Beta)*
The following **MQMD fields** are set on the AMQP message as headers, **if and only if** the value in the MQ message is not the same as the AMQP default value for that property.

- MQMD.persistence  set on AMQP message as header.durable
- MQMD.expiry  set on AMQP message as header.ttl
- MQMD.priority  set on AMQP message as header.priority

Some **MQ message properties**, if they exist, are set as AMQP headers:

- MQ message property AMQPFirstAcquirer  set on AMQP message as header.first-acquirer
- MQ message property AMQPDeliveryCount  set on AMQP message as header.delivery-count

Some **MQ message properties** are set as AMQP properties:

- AMQPUserId  set on the AMQP message as properties.user-id
- AMQPTo  set on the AMQP message as properties.to
- AMQPSsubject  set on the AMQP message as properties.subject
- AMQPReplyTo  set on the AMQP message as properties.reply-to
- AMQPContentType  set on the AMQP message as properties.content-type
- AMQPContentEncoding  set on the AMQP message as properties.content-encoding
- AMQPCreationTime  set on the AMQP message as properties.creation-time
- AMQPGroupId  set on the AMQP message as properties.group-id
- AMQPMessagesId  set on the AMQP message as properties.message-id
- AMQPGPsequence  set on the AMQP message as properties.group-sequence
- AMQPAbsoluteExpiryTime  set on the AMQP message as properties.absolute-expiry-time
- AMQPReplyToGroupId  set on the AMQP message as properties.reply-to-group-id

Finally, all **MQ message properties in the user space** (i.e. those which start usr.*) are copied into the AMQP message as application properties.
These features aren't yet available in the beta, but they're on our backlog:

- Parity with MQ Distributed Platform coverage
- Connection Authentication Rules
- Segregating AMQP/MQ Light messages from other MQ applications (see next slide)
- Enhanced Monitoring
- KnowledgeCenter documentation
- Administering shared destinations
Managing MQ Light in an MQ Environment

App group 1 publishing on /sports/football

App group 2 publishing on /sports/football

Queue manager

**Channel Attribute:**

- CHANNEL (AMQP.CHL.1)  CHLTYPE (AMQP)  TOPROOT (GROUP1.TOPIC)
- PORT (5672)

If GROUP1.TOPIC has the topic string /groups/1, messages end up on /groups/1/sports/football

- CHANNEL (AMQP.CHL.2)  CHLTYPE (AMQP)  TOPROOT (GROUP2.TOPIC)
- PORT (5673)

If GROUP2.TOPIC has the topic string /groups/2, messages end up on /groups/2/sports/football

New channel attribute for AMQP channels
Agenda

• Quick recap of MQ Light

• How MQ Light and AMQP applications work with IBM MQ (beta)

• Hybrid Messaging with MQ and Bluemix
IBM Bluemix

Bluemix is an **open-standards**, cloud-based platform for **building**, running, and managing applications.

**Build your apps, your way**
Use the most prominent compute technologies to power your app: Cloud Foundry, Docker, OpenStack.

**Scale more than just instances**
Development, monitoring, deployment, and logging tools allow the developer to run and manage the entire application.

**Extend apps with services**
A catalog of IBM, third party, and open source services allow the developer to stitch an application together quickly.

**Deploy and manage hybrid apps seamlessly**
Get a seamless dev and management experience across a number of hybrid implementations options.

**Layered Security**
IBM secures the platform and infrastructure and provides you with the tools to secure your apps.

**Flexible Pricing**
Try compute options and services for free and, when you’re ready, pay only for what you use. Pay as you go and subscription models offer choice and flexibility.
Has a “Dedicated to You” option

Single tenant hardware that’s completely dedicated to you – allowing you to satisfy regulatory & legal compliance.

• The Bluemix platform and dedicated runtimes and services sit on SoftLayer hardware that is dedicated to you

• You still have the ability to connect to all multi-tenant services in the “public” catalog

• Integrated to your LDAP for developer authentication

• Elastic capacity based on your demands.
IBM Bluemix

- Based on Pivotal® Cloud Foundry®
- Uses standard cf commands, e.g.
  - cf push <myapp>
  - cf start <myapp>
  - cf stop <myapp>
  - cf apps
  - cf create-service …
  - cf bind-service …
Applications are bound to the services they require

This is the dashboard for one of my applications. It’s bound to several Bluemix services:

- A Cloudant NoSQL database service, for storing data
- A Bluemix Secure Gateway service, for Hybrid Messaging
- A Watson Personality Insights service, for text analysis
Hybrid Messaging – Joining the 2 worlds together

- Systems of record
- Enterprise data
- 24 x 7 x 365 applications

- Systems of engagement
- Mobile
- Social
- Analytics & Watson
- Rapid development
Hybrid Messaging

Connecting the cloud with the enterprise

- **Bluemix Secure Gateway** provides a tunnel between Bluemix and your enterprise

- Gives your Bluemix apps an IP and port that routes through to a server in the enterprise

- Requires Docker in the enterprise to act as the reverse-tunnel agent
Thank You - Questions?

Related sessions:

• Introduction to MQ Light & Bluemix
  • Tuesday 1.00pm

• Repeat of this session
  • Wednesday 9.50am
Please Note

IBM’s statements regarding its plans, directions, and intent are subject to change or withdrawal without notice at IBM’s sole discretion.

Information regarding potential future products is intended to outline our general product direction and it should not be relied on in making a purchasing decision.

The information mentioned regarding potential future products is not a commitment, promise, or legal obligation to deliver any material, code or functionality. Information about potential future products may not be incorporated into any contract. The development, release, and timing of any future features or functionality described for our products remains at our sole discretion.

Performance is based on measurements and projections using standard IBM benchmarks in a controlled environment. The actual throughput or performance that any user will experience will vary depending upon many factors, including considerations such as the amount of multiprogramming in the user’s job stream, the I/O configuration, the storage configuration, and the workload processed. Therefore, no assurance can be given that an individual user will achieve results similar to those stated here.
Trademark Statement

- IBM and the IBM logo are trademarks of International Business Machines Corporation, registered in many jurisdictions. Other marks may be trademarks or registered trademarks of their respective owners.

- Microsoft, Windows, Windows NT, and the Windows logo are trademarks of Microsoft Corporation in the United States, other countries, or both.

- Java and all Java-based trademarks and logos are trademarks or registered trademarks of Oracle and/or its affiliates.

- Red Hat Enterprise Linux is a registered trademark of Red Hat, Inc. in the United States and other countries.

- Ubuntu and Canonical are registered trademarks of Canonical Ltd.

- SUSE and SLES are registered trademarks of SUSE LLC in the United States and other countries.

- Mac and OS X are trademarks of Apple Inc., registered in the U.S. and other countries.

- Other company, product and service names may be trademarks, registered marks or service marks of their respective owners.

- References in this publication to IBM products and services do not imply that IBM intends to make them available in all countries in which IBM operates.