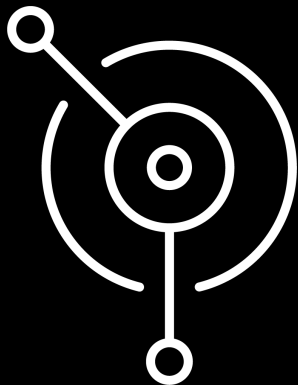


MQ Application Development Essentials & Anti-Patterns

Jamie Squibb
IBM UK

November 2019
Session **JM**





IBM **MQ**

Application Essentials: Introduction

Developer engagement

Developers can find it hard to get started with MQ

How do we make it easier for new application teams to pick up MQ and integrate with their development practices?

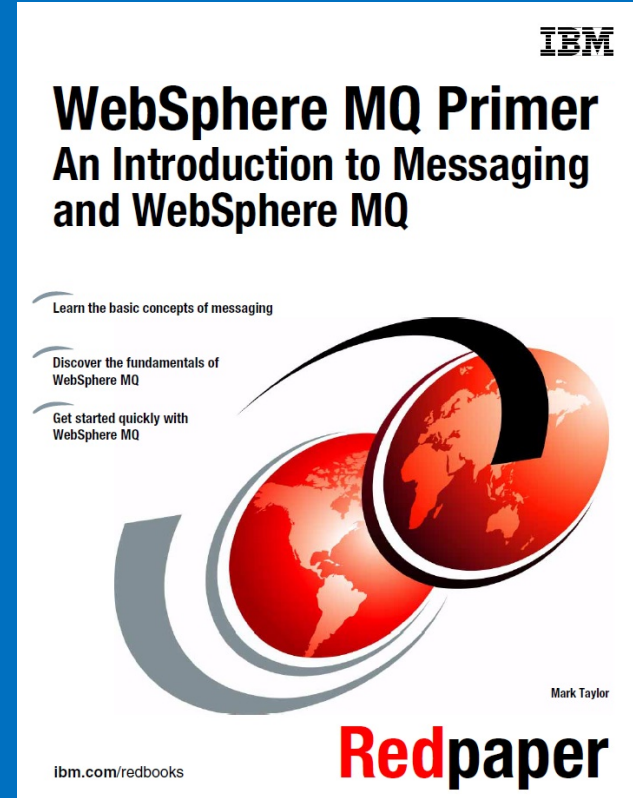
Not always been a product focus – spent more time on administration enablement

Simplifying application development and support

- Improves developer experience
- Indirectly improves the lives of administrators
- Better MQ applications = fewer headaches for administrators

Developer engagement

This is what developer engagement
used to look like



Mission statement

To enable a user instructed to use MQ for the first time, to go from zero understanding to ***running*** a sample application in a sandbox environment with a fundamental understanding of MQ concepts in 2 hours

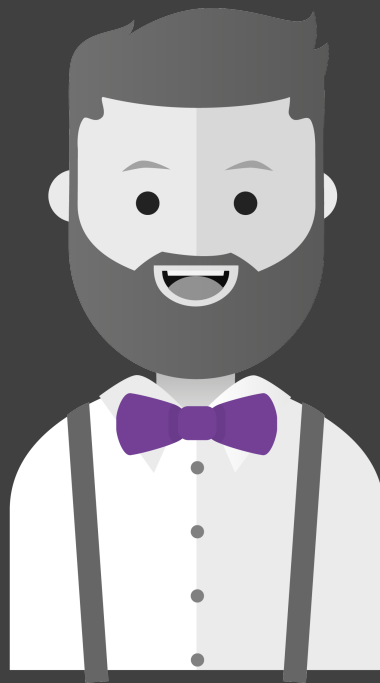
To enable an application developer, instructed to use MQ for the first time, to go from zero understanding to ***writing*** their first MQ application in the language and environment of their choice within an afternoon

developer: so what?

Just show me an API and I'll code.

Working code speaks for itself.

Ready to learn and excited to grow marketable skills.



developer: so what?

Last year, MQ celebrated its 25th birthday and so is Andre, a new application developer.

He needs to deliver a reliable MQ application for his business.



Andre

Application Developer

My job is to get
code working

I need to deliver a
new application

Skills

Java

C

Ruby

REST

Linux

responsibilities

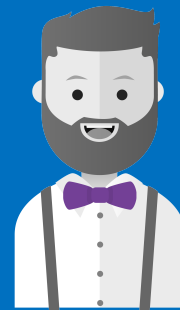
- App design & logic
- Getting new app connected to the MQ Network
- DevOps
- Incident resolution
- Getting to done!

needs

- Find information on MQ
- Core concepts of MQ
- Build applications with MQ
- Some community support
- Marketable skills

likes

- Samples, assets and code to get started (GitHub)
- Working in his preferred dev environment
- Doing stuff the *right way*
- Minimal setup
- Enjoying work
- Help when he needs it

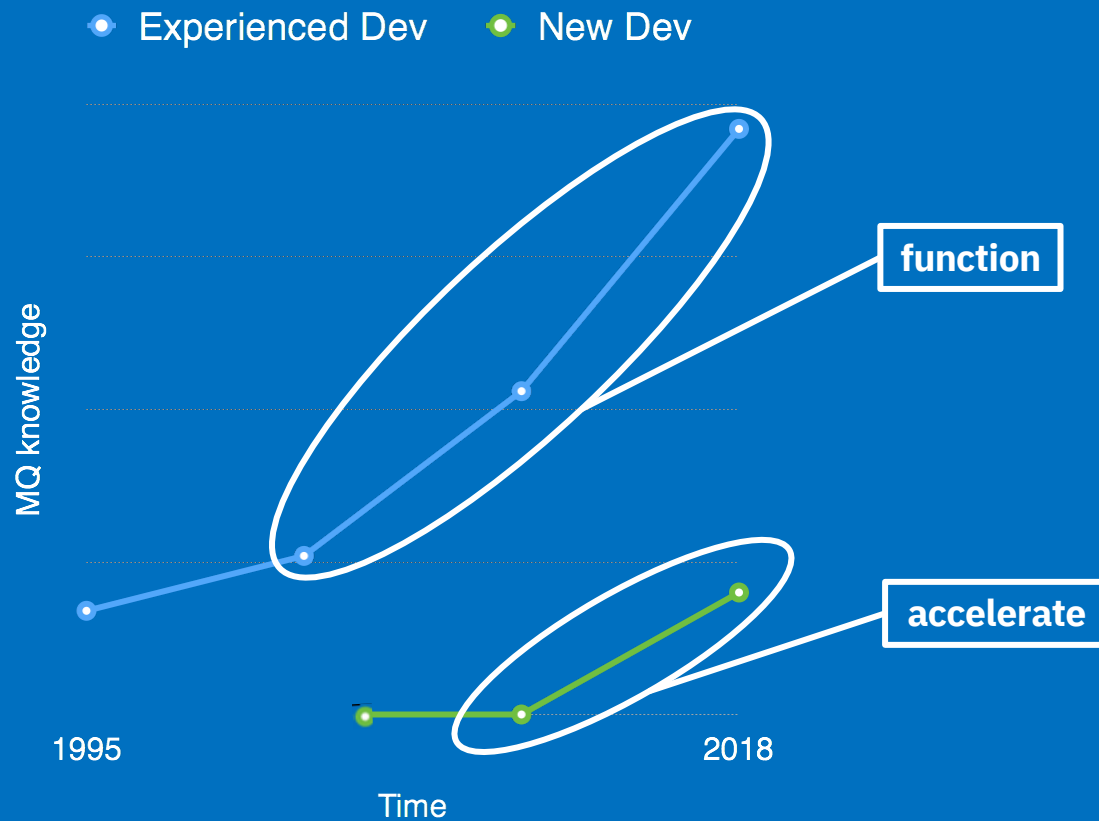




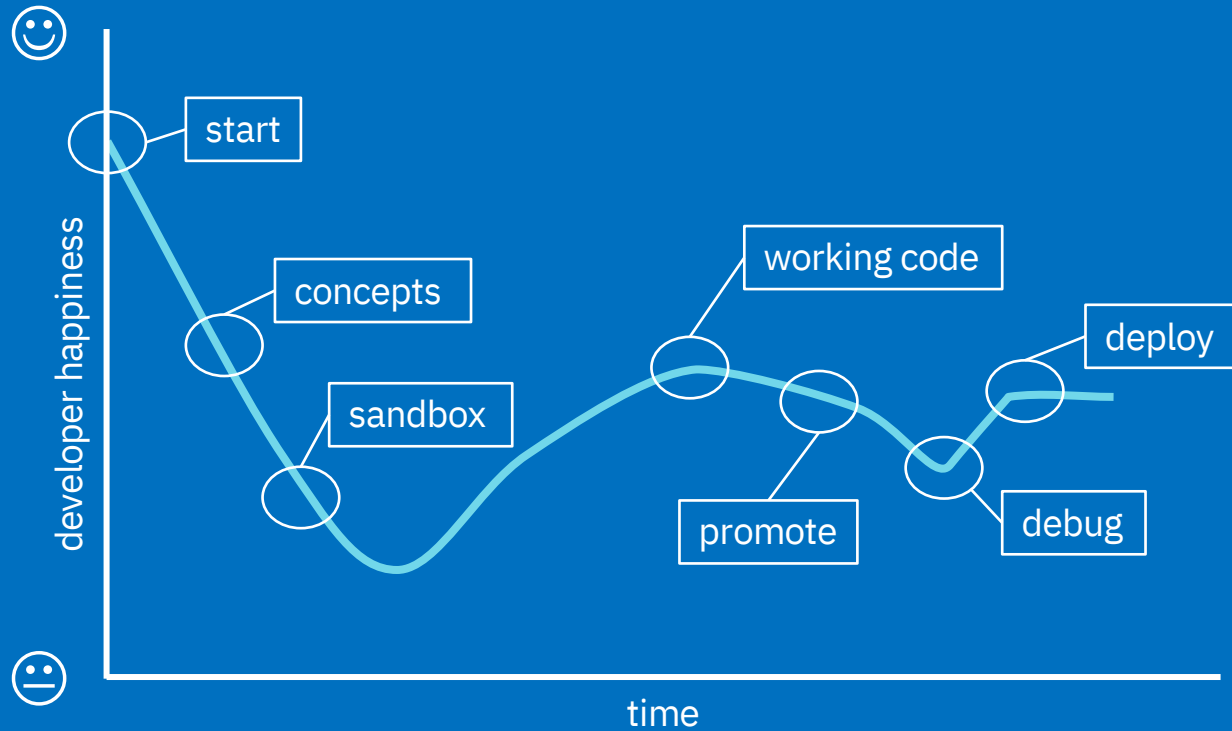
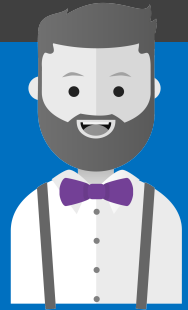
Just one thing...

**He's not used
IBM MQ before**

developer: knowledge cycle

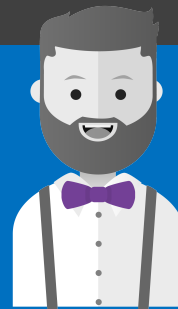
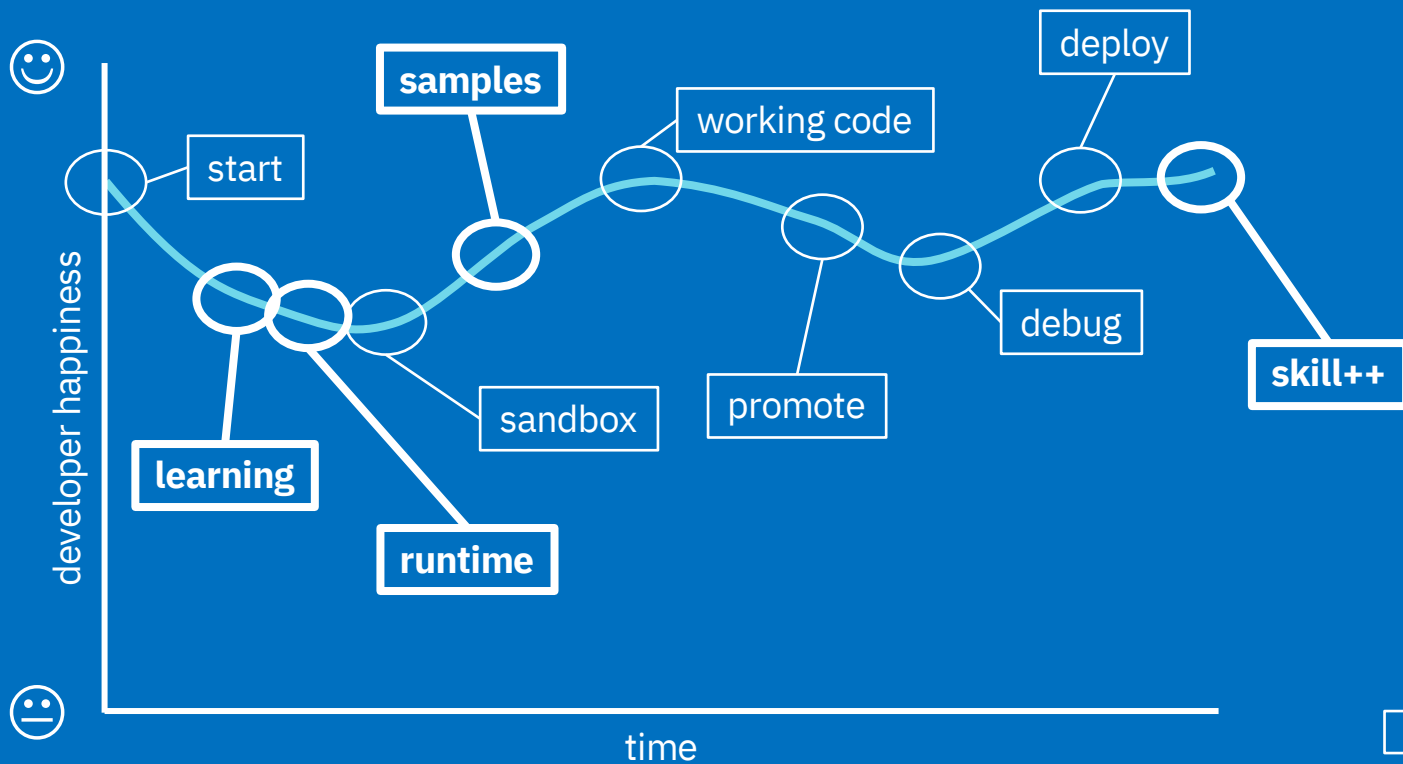


dev experience: early research



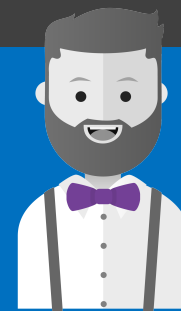
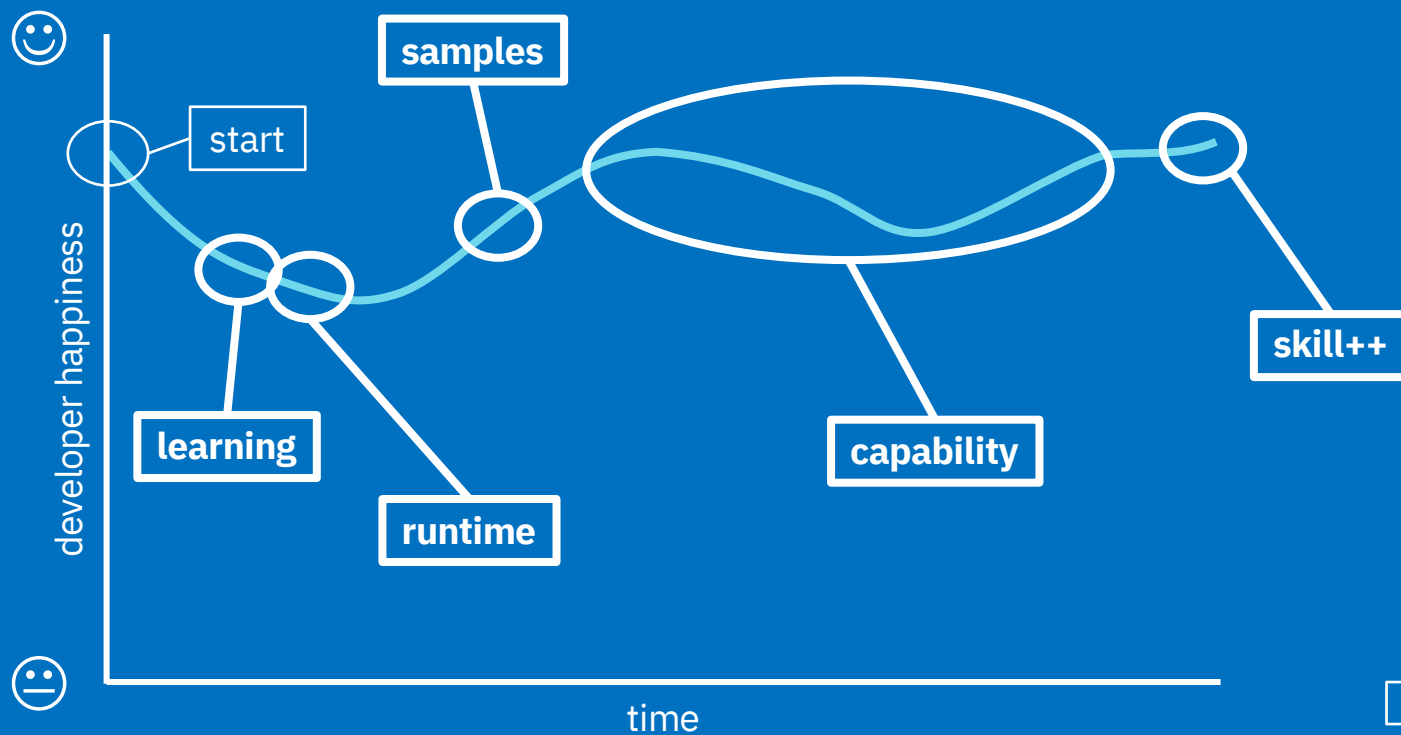
 = stages of app development

dev experience: positive direction



 = stages of app development
 = our focus areas

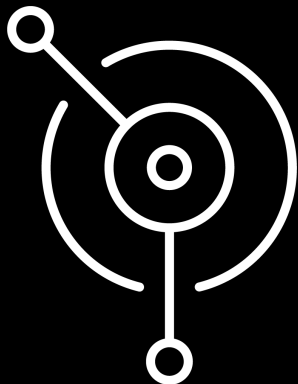
dev experience: positive direction



More than just APIs: scale, reliability, security and extensibility capabilities are critical.

Need working production ready code to succeed.

IBM MQ provides enterprise grade messaging and is a very marketable skill.



IBM **MQ**

Application Essentials: Getting Started

Welcome to messaging, made easy.

IBM MQ is the world's most powerful messaging solution.
LearnMQ will teach you how to master it.



ibm.biz/learn-mq

Finding it hard to get developers started with MQ?

Point them to:

developer.ibm.com/messaging/learn-mq

Totally new to MQ? Learn the basics

Step-by-step guides to getting up and running with MQ

Tutorials on building your applications and avoiding bad practices

The basics of MQ
With us so far? Great.

Message packages of data produced and consumed by applications.

Queuer: customizable locations to deliver messages to and store them reliably until they need to be consumed.

Queue managers: actual MQ engines, the servers that host the queues.

Channels: the way queue managers communicate with each other and with the applications.

MQ networks: loose collections of interconnected queue managers, all working together to deliver messages between applications and locations.

MQ clusters: tight groupings of queue managers, enabling higher levels of scaling and availability.

Ready, set, connect!
Connect your first application to a queue manager.

Pick your platform

MQ on Windows
A quick way to learn IBM® MQ, set up a queue manager and a sample application, and see how it works. This guide is for Windows users.

MQ on Docker
A quick way to get up and running with a queue manager and a sample application on Docker. This guide is for Docker users.

MQ on Cloud
A quick way to get up and running with a queue manager and a sample application on a cloud platform. This guide is for cloud users.

What you will learn

What you will need

Contents

1. Install Docker

2. Get the MQ in Docker image

3. Run the container from the image

4. Set up your environment

5. Connect your application to the queue manager

MQ tutorials, taking you further

Every great achievement starts with a single step. Here's a list of guides to help you get started with MQ.

Search by:

Skill level

Language

Operating system

Protected: Point-to-point with JMS

Protected: MQ Essentials

Protected: Ready, Set, Connect (Windows)

Protected: Ready, set, connect (Linux)

Point to point with JMS

Write and run your first IBM® MQ JMS application

What you will learn

What you will need

Contents

1. Point to point with JMS and IBM® MQ

2. Set up your environment

3. Connect your application to the queue manager

4. Set up your environment

5. Connect your application to the queue manager

MQ tutorials, taking you further

Every great achievement starts with a single step.
Here's a set of guided tutorials that provides you with the tools to master MQ.



MQ Essentials (5-10 mins)

Get a queue manager (10-15 mins)

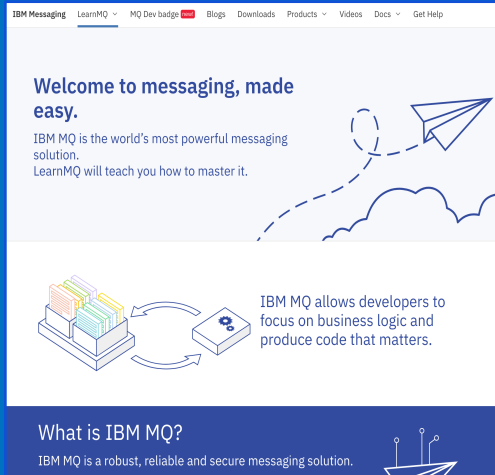
Hello world (10-15 mins)

25 years in 25 minutes!

LearnMQ

Developer Essentials Badge and challenge

Samples and patterns



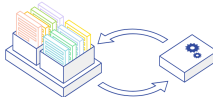
IBM Messaging LearnMQ MQ Dev badge Blogs Downloads Products Videos Docs Get Help

Welcome to messaging, made easy.

IBM MQ is the world's most powerful messaging solution. LearnMQ will teach you how to master it.

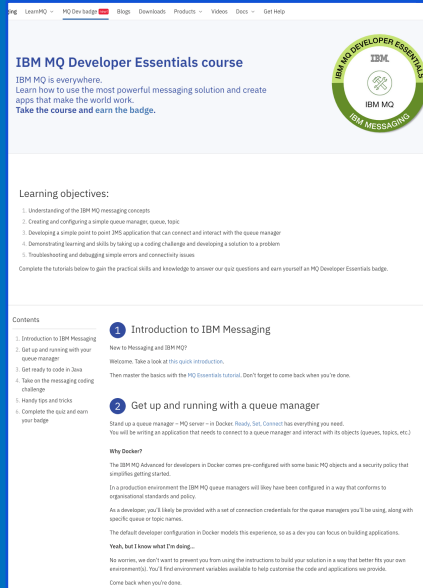


IBM MQ allows developers to focus on business logic and produce code that matters.



What is IBM MQ?

IBM MQ is a robust, reliable and secure messaging solution.



IBM MQ Developer Essentials course

IBM MQ is everywhere. Learn how to use the most powerful messaging solution and create apps that make the world work. Take the course and earn the badge.



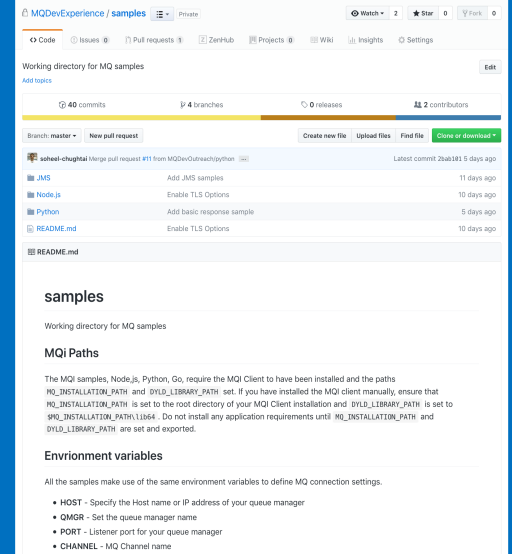
Learning objectives:

1. Understanding the IBM MQ messaging concepts
2. Creating and configuring a simple queue manager, queue, topic
3. Developing a simple point to point JMS application that can connect and interact with the queue manager
4. Demonstrating learning and skills by taking on a coding challenge and developing a solution to a problem
5. Troubleshooting and debugging simple errors and connectivity issues

Complete the tutorials below to gain the practical skills and knowledge to answer our quiz questions and earn yourself an IBM Developer Essentials badge.

Contents

1. Introduction to IBM Messaging
 - 1. Introduction to IBM Messaging
 - 2. Set up and running with your queue manager
 - 3. Get ready to code in Java
 - 4. Take on the messaging coding challenge
 - 5. Ready to go to work
 - 6. Complete the quiz and earn your badge
2. Get up and running with a queue manager
 - 1. Introduction to IBM Messaging
 - 2. Set up and running with your queue manager
 - 3. Get ready to code in Java
 - 4. Take on the messaging coding challenge
 - 5. Ready to go to work
 - 6. Complete the quiz and earn your badge



MQDevExperience / samples Private Watch 2 Star 0 Fork 0

Code Issues Pull requests ZenHub Projects Wiki Insights Settings

Working directory for MQ samples

Add topics

40 commits 4 branches 0 releases 2 contributors

Branch: master New pull request Create new file Upload files Find file Clone or download

File	Commit	Latest commit	Time ago
sample-challenge	Large pull request #11 from MQDevExperience	29ab181	5 days ago
JMS	Add JMS samples		11 days ago
Node.js	Enable TLS Options		10 days ago
Python	Add basic response sample		5 days ago
README.md	Enable TLS Options		10 days ago

README.md

samples

Working directory for MQ samples

MQI Paths

The MQI samples, Node.js, Python, Go, require the MQI Client to have been installed and the paths `MQ_INSTALLATION_PATH` and `QVLS_LIBRARY_PATH` set. If you have installed the MQI client manually, ensure that `MQ_INSTALLATION_PATH` is set to the root directory of your MQI client installation and `QVLS_LIBRARY_PATH` is set to `MQ_INSTALLATION_PATH\lib64`. Do not install any application requirements until `MQ_INSTALLATION_PATH` and `QVLS_LIBRARY_PATH` are set and exported.

Environment variables

All the samples make use of the same environment variables to define MQ connection settings.

- **HOST** - Specify the Host name or IP address of your queue manager
- **QMGR** - Set the queue manager name
- **PORT** - Listener port for your queue manager
- **CHANNEL** - MQ Channel name

<http://ibm.biz/mq-dev-patterns>

(No longer need to install the product just to see sample code!)

LearnMQ tutorials

A collage of screenshots from the IBM MQ documentation website. The top row shows the 'Welcome to messaging, made easy.' page, 'MQ Essentials' (Getting started with IBM MQ), and 'Point to point with JMS'. The middle row features 'MQ with TLS', 'MQ with Microclimate', and 'Slow MQ app, lost messages and high CPU? Three easy ways to improve your MQ application'. The bottom row includes 'MQ messaging REST API tutorial', 'What you will learn' (for TLS and Microclimate), and 'What you will need' (for TLS and Microclimate). The right side of the collage shows a 'Contents' page for the REST API tutorial and a 'What you will learn' page for the REST API. The bottom right corner shows a 'What you will need' page for the REST API. The screenshots are arranged in a grid-like fashion, with some overlapping, creating a comprehensive overview of the documentation's structure and content.

LearnMQ

MQ Dev badge

Downloads

Products

Videos

Docs

Get Help

IBM MQ Developer Essentials

IBM

IBM MQ

IBM MESSAGING

IBM MQ Developer Essentials course

IBM MQ is everywhere.
Learn how to use the most powerful messaging solution and create apps that make the world work.
Take the course and earn the badge.

Learning objectives:

- 1. Understanding of the IBM MQ messaging concepts
- 2. Creating and configuring a simple queue manager, queue, topic
- 3. Developing a simple point to point JMS application that can connect and interact with the queue manager
- 4. Demonstrating learning and skills by taking up a coding challenge and developing a solution to a problem
- 5. Troubleshooting and debugging simple errors and connectivity issues

Complete the tutorials below to gain the practical skills and knowledge to answer our quiz questions and earn yourself an MQ Developer Essentials badge.

Contents

- 1. Introduction to IBM Messaging
- 2. Get up and running with your queue manager
- 3. Get ready to code in Java
- 4. Take on the messaging coding challenge
- 5. Handy tips and tricks
- 6. Complete the quiz and earn your badge

1

Introduction to IBM Messaging

New to Messaging and IBM MQ?

Welcome. Take a look at [this quick introduction](#).

Then master the basics with the [MQ Essentials tutorial](#). Don't forget to come back when you're done.

2

Get up and running with a queue manager

Stand up a queue manager – MQ server – in Docker. [Ready](#), [Set](#), [Connect](#) has everything you need.

You will be writing an application that needs to connect to a queue manager and interact with its objects (queues, topics, etc)

Why Docker?

The IBM MQ Advanced for developers in Docker comes pre-configured with some basic MQ objects and a security policy that simplifies getting started.

In a production environment the IBM MQ queue managers will likely have been configured in a way that conforms to organisational standards and policy.

As a developer, you'll likely be provided with a set of connection credentials for the queue managers you'll be using, along with specific queue or topic names.

The default developer configuration in Docker models this experience, so as a dev you can focus on building applications.

Yeah, but I know what I'm doing...

No worries, we don't want to prevent you from using the instructions to build your solution in a way that better fits your own environment(s). You'll find environment variables available to help customise the code and applications we provide.

Come back when you're done.

IBM Messaging LearnMQ - MQ Dev badge Blogs Downloads Products - Videos Docs - Get Help

Welcome to messaging, made easy.

IBM MQ is the world's most powerful messaging solution.

LearnMQ will teach you how to master it.

IBM Messaging LearnMQ - MQ Dev badge Blogs Downloads Products - Videos Docs - Get Help

MQ Essentials

Getting started with IBM MQ

Blog Downloads Products - Videos Docs - Get Help

Ready, set, connect!

Connect a simple MQ application to a queue manager.

Pick your platform

As you learn MQ (prepare to be a team application, you need both the queue manager (MQ server) and the client application (MQ client). In this tutorial, you get to play with both. There are many ways to get hold of the MQ server. You can install a ready-to-run version (pre-installed), or you can install a queue manager in your cloud. For all the details on this page, check out our example of getting started with MQ.

Click to choose a tutorial: [native MQ installation on Windows](#), [MQ on Docker](#) or [MQ in IBM Cloud](#).

MQ on Windows

Install IBM MQ, set up a queue manager and install a client demo application, all on one Windows environment.

45 minutes

MQ on Docker

Run and connect two Docker containers, one with an MQ queue manager and one with the MQ client demo application.

30 minutes

MQ on Cloud

Deploy an IBM MQ queue manager in IBM Cloud and connect it with a client demo application here running on Linux, Windows or Mac.

15 minutes

What you will learn

- 1. A bit about IBM queue managers
- 2. A few basic MQ queries
- 3. The basics of point-to-point messaging

What you will need

- 1. Docker
- 2. The latest IBM Docker image from Docker Hub
- 3. IBM MQ point-to-point demo

Contents

1. Install Docker
2. Get the MQ in Docker image
3. Run the container from the image
4. Run the demo application in Docker
5. Put and get messages to and from a queue

"What do I need to start this tutorial?"

Just your laptop.

1. Install Docker

2. Get the MQ in Docker image

Containers are run from images and images are built from a specification listed in a Dockerfile. We will use a pre-built IBM MQ server image from Docker Hub. For more details on installing Docker, instructions are here. If your Docker system is setup Docker or Docker engine, you need to install Docker before installing the latest docker or version. Once you've installed Docker, come back to continue with the tutorial.

The image is a collage of three screenshots from the IBM MQ Developer website. The top screenshot shows the 'Point to point with JMS' section, highlighting the 'MQ Badge developer challenge' and a link to 'Join the dots, show off your messaging skills'. The middle screenshot shows the 'What you will learn' section, listing topics like 'Publish/subscribe basics', 'Request/response basics', and 'Transactional messages'. The bottom screenshot shows the 'Contents' section, listing topics like 'The challenge', 'Architecture overview', and 'Request/response', along with a '1 The challenge' section header.

The screenshot displays the GitHub interface for the 'ApacheMQ' repository. At the top, navigation links include 'BM Messaging', 'LearnMQ', 'MQ Dev badge', 'Blogs', 'Downloads', 'Products', 'Videos', 'Docs', and 'Get Help'. The main heading is 'MQ cheat sheet for developers' with the subtitle 'Tips and tricks for easy debugging'. Below this, a section titled 'What you will learn' lists three points: common errors, where to find debug info, and how to find MQ object info. The repository stats show 1 pull request, 3 issues, marketplace, and explore options. The file browser shows a directory named '-badge-sample' with 1 branch, 0 releases, 3 contributors, and Apache-2.0 license. A table of commits follows, listing recent changes like 'Add source files to GenerateEvents.jar' and 'Remove Transactions from ModelAnswer'. The bottom section introduces the 'Developer challenge' as a starting point for building a ticket reseller system.

Why a Badge?

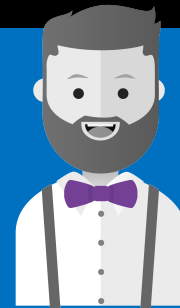
Sets a modest but achievable bar for a new developer to attain in a short period of time

Guarantees a consistent set of terms and concepts for all new MQ devs

Doesn't require weeks to achieve

Gives MQ administrators:

- a single place to point new developers at
- a known starting point once developers have completed it



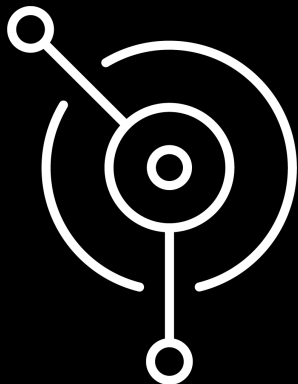
IBM MQ Developer Essentials course

IBM MQ is everywhere.

Learn how to use the most powerful messaging solution and create apps that make the world work.

Take the course and earn the badge.





IBM **MQ**

Application Essentials: Recent Updates

Demonstrating the simplicity of MQ

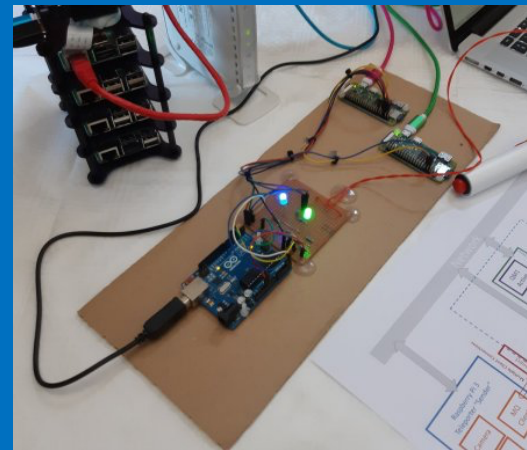
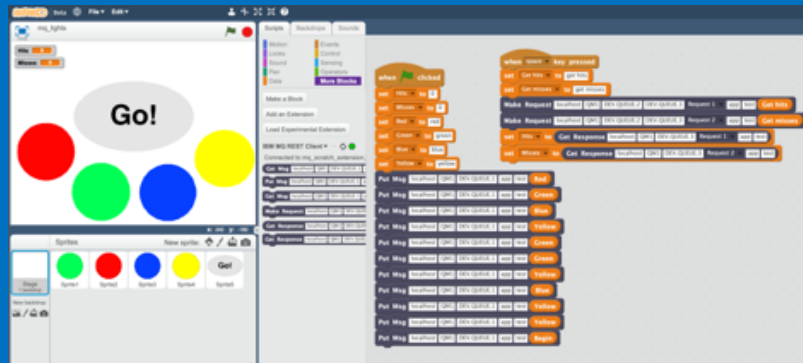
There's nothing like flashing lights and wires to grab people's attention. We want everyone to know how easy it is to write messaging applications and how powerful MQ is in supporting them

Ever tried **Scratch**, a graphical way to code, aimed at kids but ideal to show how easily asynchronous messaging can improve your applications with an MQ plugin

ibm.biz/ibmmq-scratch

Heard of the **Raspberry PI**? You think MQ is a heavyweight solution? We run an HA pair of queue managers on two \$5 Raspberry PI Zeros!

ibm.biz/ibmmq-pi



Multiple APIs and protocols

IBM MQ supports multiple APIs and multiple client protocols. Both proprietary and open.

APIs: MQI, JMS, MQ Light, REST ...

Protocols: MQ, AMQP, MQTT, HTTP

These support a wide range of application styles, from the simplest of messaging needs through to the richest

MQ is the transport; messages produced from any API or protocol can be received by any other API or protocol



APIs

MQI

Exposes the full set of MQ capabilities
Uses the MQ protocol

JMS 2.0

Supports the full JMS API for use in many JSE or JEE environments
Uses the MQ protocol

MQ Light

A simple pub/sub messaging API
Uses the AMQP 1.0 protocol

Protocols

MQ

MQ's highly reliable and performant messaging protocol

MQTT

Supports the open standard MQTT API and protocol
Open source Eclipse Paho clients

AMQP

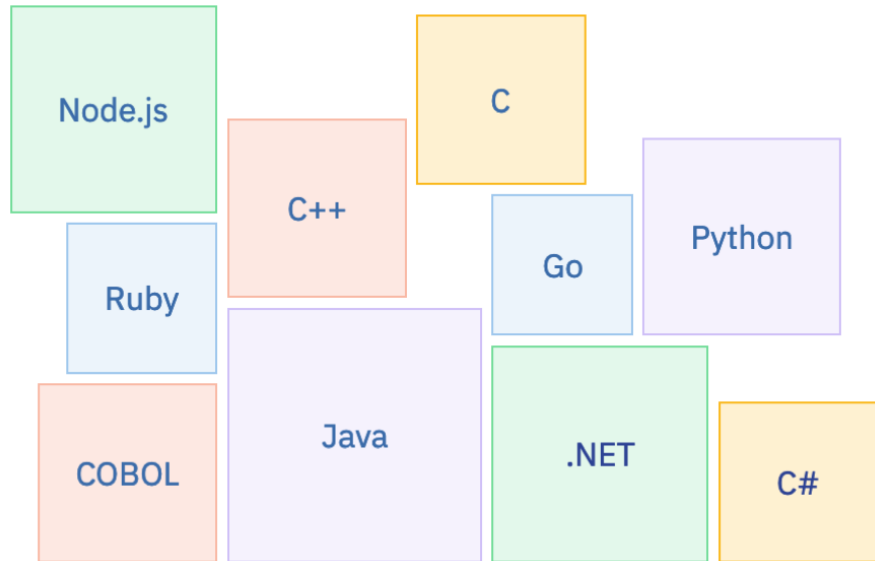
Support for AMQP 1.0 enables support for open source clients such as Qpid Proton

REST

A very simple but secure messaging API over REST

...

MQ speaks your language



Recent updates

APIs &
Protocols

REST

Go

Node.js

Maven

MQLight

Spring

Docker

LTS & CD

Developing applications

Build your applications simply, with no need for an MQ installation

Pull Java directly from the Maven repository since MQ 9.0.4 CD

MQ 9.1.1 CD added the **SDK** to the MQ redistributable client

The redistributable client is now available directly, no need to log into IBM

ibm.biz/mqclientdownload

Develop your applications on the platform of your choice with the addition of the MacOS version of the 9.1.1 MQ client and SDK for Developers

ibm.biz/mqmacos

(The MQ for MacOS toolkit includes runmqsc)



New MQI languages

The MQI API exposes the richest MQ messaging capabilities

Traditionally mostly used by C and COBOL

Other bindings have been available

New interfaces available for GO and Node.js designed around full MQI function

Provided as as-is open source, let us know if there's a need to take these further

Built on the C MQI library

The screenshot shows the GitHub repository page for `ibm-messaging / mq-golang`. The repository has 12 watchers, 31 stars, and 9 forks. It contains 31 commits, 1 branch, 0 releases, 1 contributor, and is licensed under Apache-2.0. The repository is titled "Calling IBM MQ from Go applications". The file list includes `cmd`, `ibmmq`, `mqmetric`, `CLA.md`, `LICENSE`, and `README.md`. The latest commit is `c44b74e` on 20 Jul.

The screenshot shows the GitHub repository page for `ibm-messaging / mq-mqi-nodejs`. The repository has 5 watchers, 0 stars, and 0 forks. It contains 4 commits, 1 branch, 0 releases, 1 contributor, and is licensed under Apache-2.0. The repository is titled "Calling IBM MQ from JavaScript - an MQI wrapper". The file list includes `lib`, `samples`, `CLA.md`, `LICENSE`, and `README.md`. The latest commit is `8976669` 4 days ago.

New MQI languages - examples

Intent to make it easy to write for MQ from newer languages and environments

In a natural style for the language

e.g. strings, not padded-fixed-lengths

Helps you to develop apps wherever you need

For **Node.js JavaScript** see
github.com/ibm-messaging/mq-mqi-nodejs

or `require('ibmmq')` from NPM

For **Go** bindings see
github.com/ibm-messaging/mq-golang

Both include sample programs to help get started

```
mqmd = new mq.MQMD();
pmo = new mq.MQPMO();
msg = "Hello from Node";

mq.Put(hObj, mqmd, pmo, msg, function(err) {
    if (err) {
        console.log(err);
    }
});
```

```
mqmd := ibmmq.NewMQMD()
pmo := ibmmq.NewMQPMO()
mqmd.Format = "MQSTR"
msg := "Hello from Go"

buffer := []byte(msg)
err = qObject.Put(mqmd, pmo, buffer)

if err != nil {
    fmt.Println(err)
}
```

Availability of runtimes

Docker container with full MQ server function

Creates sample objects for developers
Queues, topics, userids etc

See <https://hub.docker.com/r/ibmcom/mq/>

Redistributable Client packages now easier to access

To make it easy for developers to package standalone applications
Perhaps in a container

public.dhe.ibm.com/ibmdl/export/pub/software/websphere/messaging/mqdev/redis

REST messaging

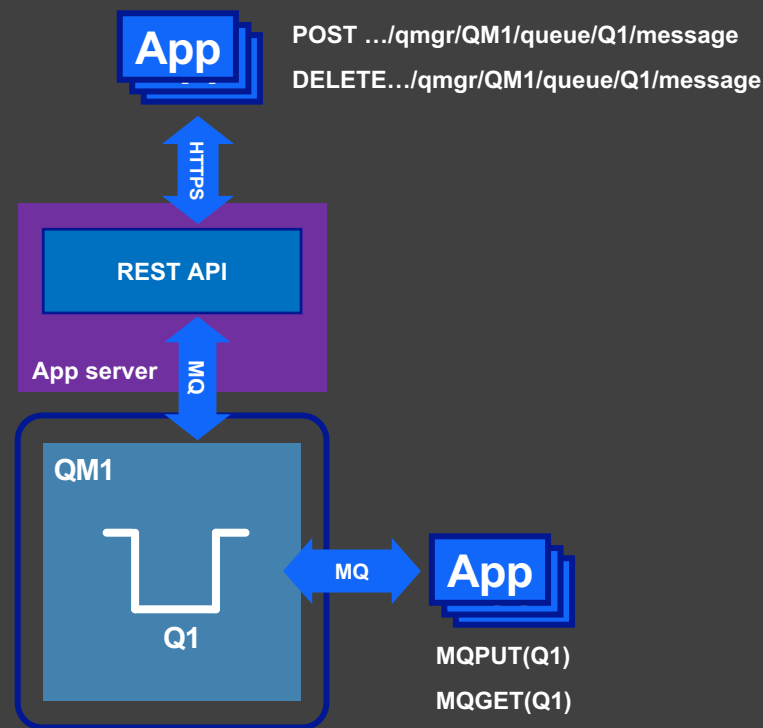
Many users just want the simplest way to get messages in and out of an MQ system.

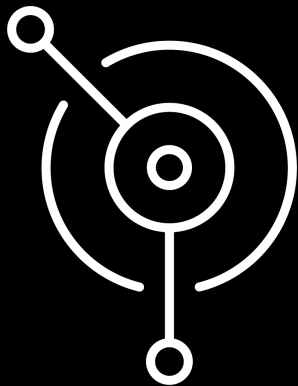
A RESTful API gives you just that. Easily enabling messaging from just about any environment with no need for an MQ client or platform/language limitations.

The new HTTP server support in MQ 9.0.x provides the platform for a properly integrated REST API solution.

Simple synchronous point-to-point support.

MQ support for z/OS Connect also provides a similar mechanism.





IBM **MQ**

Application Essentials: Anti-Patterns

Application design considerations

Application design can affect:

- Performance
- Availability
- Reliability
- Scalability

... for the application, but potentially also a queue manager

Anti-Pattern: Repeatedly connecting

Common development approach:

Step 1: Write code to get/send a message

Step 2: Repeatedly call code written in step 1

Common pitfalls:

- ✗ Repeatedly connect and disconnect
- ✗ Repeatedly open and close queues/topics

Impact:

- Poor performance for application
- Greater overhead for queue manager

Solution:

- Reuse session, context and object handles
- Use LLA to pre-load SCSQAUTH/SCSQANLx for local application connections on z/OS

```
for (int i = 1; i <= 5000; i++)
{
    context = cf.createContext();
    destination = context.createQueue("queue://" + QUEUE_NAME);
    TextMessage message =
        context.createTextMessage("This is message number " + i + ".\n");
    producer = context.createProducer();
    producer.send(destination, message);
    context.close();
}
```



```
context = cf.createContext();
destination = context.createQueue("queue://" + QUEUE_NAME);
producer = context.createProducer();

for (int i = 1; i <= 5000; i++)
{
    TextMessage message =
        context.createTextMessage("This is message number " + i + ".\n");
    producer.send(destination, message);
}

context.close();
```

Anti-Pattern: Polling

Common development approach:

Step 1: Write code to get a message

Step 2: If no message available then call again

Common pitfalls:

- ✗ Rapidly and repeatedly fail to get a message because one is not available

Impact:

- High CPU usage for application
- Greater overhead for queue manager

Solution:

- Poll less often or wait longer for messages
- Use event driven messaging (async-consume)
 - Ideal when getting from multiple queues

```
while (true)
{
    // Attempt to get a message, waiting for up to 1ms
    Message receivedMessage = consumer.receive(1);

    ... process message ...
}
```



```
while (true)
{
    // Attempt to get a message, waiting indefinitely
    Message receivedMessage = consumer.receive();

    ... process message ...
}
```

OR

```
context = cf.createContext();
destination = context.createQueue("queue://" + QUEUE_NAME);
consumer = context.createConsumer();

MessageListener ml = new MyMessageListener();
consumer.setMessageListener(ml);

-----

public class MyMessageListener implements MessageListener
{
    public void onMessage(Message message)
    {
        ... process message ...
    }
}
```

Anti-Pattern: Optimistic programming

Common development approach:

Step 1: Write code to send/receive messages

Step 2: Declare success → move on to next task

Common pitfalls:

- ✗ Forget to write code to handle errors or exceptions

Potential impact:

- Messages are discarded or processed incorrectly
- Application is unreliable, performs badly, or crashes

Solution:

- Check return codes and handle exceptions

```
TextMessage message =  
    context.createTextMessage("This important message MUST be sent");  
  
producer.send(destination, message);  
  
System.out.println("Sent message!");
```



```
try  
{  
    TextMessage message =  
        context.createTextMessage("This important message MUST be sent");  
  
    producer.send(destination, message);  
  
    System.out.println("Sent message!");  
}  
catch (Exception ex)  
{  
    Throwable causeex = ex.getCause();  
  
    if (causeex instanceof MQException)  
    {  
        ...  
    }  
  
    ...  
}
```

Anti-Pattern: Long-running UOWs

Causes:

- ✗ Processing large numbers of messages within a transaction
- ✗ Forgetting to commit and start a new transaction
- ✗ Incorrect sync-point option – default is different on z/OS

Potential impact:

- Increased resource requirements for the queue manager (storage, locks, recovery log)
- Can cause a queue manager outage in extreme circumstances
- Application errors if max UOW size reached

Related notes:

- ✓ Keep transactions as small as possible – do the actions really need to be performed atomically?
- ✓ Usually best to use a sync-point when processing persistent messages
- ✓ Cheaper to process multiple persistent messages in a single UOW – no need to force the log after each put or get.
- ✓ UOW size limited by queue manager MAXUMSGS parameter
- ✓ Remember to check whether transactions are committed successfully

Anti-Pattern: Exclusivity

Applications can open a queue to get messages exclusively

- ✓ Can be ideal when sequence is important

Potential impact:

- ✗ Harder to scale application
- ✗ Harder to exploit shared queues on z/OS

Can another technique be used instead, such as partitioning using a correlation ID?

Anti-Pattern: Hard-coded values

Avoid hard-coding queue manager, queue and topic names

Potential impact:

- X Harder to reuse application
- X Harder to scale application
- X Harder to promote from dev → prod

Anti-Pattern: Incorrect persistence

Persistent messages are ideal for:

- ✓ Simplifying application logic
- ✓ Auditability

However...

- Persistent messages require additional processing for recovery
- Performance can be reduced

If an application can detect and recover from lost messages in the event of a failure then persistence might not be necessary.

Other considerations

Long-running applications might consume unnecessary system resources

- Triggering an application when messages arrive might be more efficient

Is the message encoding important?

- If so, good idea to request data conversion when getting messages

Be explicit about what options are required – be careful about platform-specific default values

Notices and disclaimers

© 2019 International Business Machines Corporation. No part of this document may be reproduced or transmitted in any form without written permission from IBM.

U.S. Government Users Restricted Rights — use, duplication or disclosure restricted by GSA ADP Schedule Contract with IBM.

Information in these presentations (including information relating to products that have not yet been announced by IBM) has been reviewed for accuracy as of the date of initial publication and could include unintentional technical or typographical errors. IBM shall have no responsibility to update this information. This document is distributed “as is” without any warranty, either express or implied. In no event, shall IBM be liable for any damage arising from the use of this information, including but not limited to, loss of data, business interruption, loss of profit or loss of opportunity. IBM products and services are warranted per the terms and conditions of the agreements under which they are provided.

IBM products are manufactured from new parts or new and used parts. In some cases, a product may not be new and may have been previously installed. Regardless, our warranty terms apply.”

Any statements regarding IBM's future direction, intent or product plans are subject to change or withdrawal without notice.

Performance data contained herein was generally obtained in a controlled, isolated environments. Customer examples are presented as illustrations of how those

customers have used IBM products and the results they may have achieved. Actual performance, cost, savings or other results in other operating environments may vary.

References in this document to IBM products, programs, or services does not imply that IBM intends to make such products, programs or services available in all countries in which IBM operates or does business.

Workshops, sessions and associated materials may have been prepared by independent session speakers, and do not necessarily reflect the views of IBM. All materials and discussions are provided for informational purposes only, and are neither intended to, nor shall constitute legal or other guidance or advice to any individual participant or their specific situation.

It is the customer's responsibility to insure its own compliance with legal requirements and to obtain advice of competent legal counsel as to the identification and interpretation of any relevant laws and regulatory requirements that may affect the customer's business and any actions the customer may need to take to comply with such laws. IBM does not provide legal advice or represent or warrant that its services or products will ensure that the customer follows any law.

Notices and disclaimers continued

Information concerning non-IBM products was obtained from the suppliers of those products, their published announcements or other publicly available sources. IBM has not tested those products about this publication and cannot confirm the accuracy of performance, compatibility or any other claims related to non-IBM products. Questions on the capabilities of non-IBM products should be addressed to the suppliers of those products. IBM does not warrant the quality of any third-party products, or the ability of any such third-party products to interoperate with IBM's products. IBM expressly disclaims all warranties, expressed or implied, including but not limited to, the implied warranties of merchantability and fitness for a purpose. The provision of the information contained herein is not intended to, and does not, grant any right or license under any IBM patents, copyrights, trademarks or other intellectual property right.

IBM, the IBM logo, ibm.com and [names of other referenced IBM products and services used in the presentation] are trademarks of International Business Machines Corporation, registered in many jurisdictions worldwide. Other product and service names might be trademarks of IBM or other companies. A current list of IBM trademarks is available on the Web at "Copyright and trademark information" at: www.ibm.com/legal/copytrade.shtml.

Please submit your session feedback!

- Do it online at <http://conferences.gse.org.uk/2019/feedback/JM>
- This session is JM



1. What is your conference registration number?

💡 This is the three digit number on the bottom of your delegate badge

--

2. Was the length of this presentation correct?

💡 1 to 4 = "Too Short" 5 = "OK" 6-9 = "Too Long"

1 2 3 4 5 6 7 8 9

3. Did this presentation meet your requirements?

💡 1 to 4 = "No" 5 = "OK" 6-9 = "Yes"

1 2 3 4 5 6 7 8 9

4. Was the session content what you expected?

💡 1 to 4 = "No" 5 = "OK" 6-9 = "Yes"

1 2 3 4 5 6 7 8 9