Connecting MQ to the World: New Bridges

Mark Taylor
marke_taylor@uk.ibm.com
IBM Hursley
IBM MQ – Bridges

**Introduction**

- MQ has always included links to other application technologies
  - Not just a set of APIs
  - Not always devolved to MessageBroker/IIB

- For example
  - CICS, IMS Bridges
  - SAP R/3 Link (obsolete)
  - Resource Adapter for JEE
  - AMQP

- Now has product-provided bridges to two new environments:
  - Salesforce
  - Blockchain

- And open-source connectors for **Kafka**
Objectives

- Provide simple way for MQ apps to integrate with these new systems
- Make use of existing MQ infrastructure
- Build on MQ skills
- Complement capabilities in other products such as IIB
MQ BRIDGE FOR BLOCKCHAIN
WHAT IS BLOCKCHAIN
IBM MQ – Bridges

Problem …

- Participant A’s records
- Insurer records
- Participant B’s records
- Bank records
- Regulator records
- Auditor records

… inefficient, expensive, vulnerable
A shared, replicated, permissioned ledger …

… with consensus, provenance, immutability and finality
IBM MQ – Bridges

**Shared ledger**

- Shared between participants
- Participants have own copy through replication
- Permissioned, so participants see only appropriate transactions
- THE shared system of record
Hyperledger: A Linux Foundation project

- A collaborative effort created to advance cross-industry blockchain technologies for business

- Announced December 2015, now around 150 members

- Open source, open standards, open governance

- Five frameworks and three tools projects

- IBM is a premier member of Hyperledger

www.hyperledger.org
IBM MQ – Bridges

Hyperledger Fabric: Distributed ledger platform

- An implementation of blockchain technology that is a foundation for developing blockchain applications
- Emphasis on ledger, smart contracts, consensus, confidentiality, resiliency and scalability.
- V1.0 released July 2017
  - 159 developers from 27 organizations
  - IBM is one contributor of code, IP and development effort to Hyperledger Fabric

http://hyperledger-fabric.readthedocs.io/
THE MQ BRIDGE TO BLOCKCHAIN
IBM MQ – Bridges

Blockchain Bridge

• MQ is a natural fit to connect existing business transaction systems to remote intra-business ledger services

• Asynchronous request/reply MQ message flow for applications to request information from Blockchain ("what is the value of the balance on this account") over MQ queues
Blockchain: holds details of all vehicles for country

- Manufacturer adds vehicles
- Importer: adds vehicles
- Dealer: Registers vehicle, updates ownership
- Recycle/Scrap removes vehicles

IBM MQ – Bridges
Blockchain: holds details of all vehicles for country

Modified Hybrid Business Process

Tax Records

Tax Due process

Send Reminder process

MQ Bridge

Excise Authority Query current Owner

Reminder

MQ Bridge

Dealer: Registers vehicle, updates ownership

Recycle/Scrap removes vehicles

Tax Due process

Send Reminder process

Reminder

Tax Records

Manufacturer adds vehicles

Importer: adds vehicles

IBM MQ – Bridges

Overview

• The bridge allows an MQ app to query or update data held in a Hyperledger Fabric system
  – Without needing to know anything about the ledger configuration or controls
  – No special APIs

• Support:
  – Only supported for connections to a queue manager with "Advanced" license
    • Including z/OS VUE Advanced, MQ Appliance
  – Bridge itself supported only on xLinux and z/OS

• Initial release in 9.0.3
  – Modified and extended as Hyperledger has changed
  – Likely to change further as Hyperledged evolves

• V9.1 bridge based on Hyperledger Composer
  – A simplified, "business-level" view of resources
IBM MQ – Bridges

Initial release

Queue Manager
- Input Queue
- Reply Queue

MQ Bridge to Blockchain
- parse request, submit query
- parse response, build, return reply message
- MQ JMS Client
- Fabric Interface

Hyperledger Fabric
- Blockchain

User Application
- Request (JSON)
- Reply (JSON)
IBM MQ – Bridges

V9.1

Queue Manager

Input Queue

Reply Queue

Reply (JSON)

MQ Bridge to Blockchain

parse request, submit query

parse response, build, return reply message

MQ JMS Client

REST

User Application

Request (JSON)

Hyperledger Composer

Composer

Blockchain

Input Queue 

Reply Queue

Reply (JSON)

User Application

MQ Bridge to Blockchain

parse request, submit query

parse response, build, return reply message

MQ JMS Client

REST

Composer

Blockchain
IBM MQ – Bridges

Installation/configuration

• Bridge package is shipped as installable fileset for xLinux, and as part of z/OS Adv product
  – Includes bridge code, sample code and definitions, and prereq 3rd party jars

• Config asks for connection information to MQ and to Blockchain
  – For Blockchain, need address of REST server, a userid and password
  – For MQ, needs client connection information, optionally userid and password
  – TLS configuration

• Configure qmgr with necessary queues and security

• Can then start in normal runtime mode
  – Connections made to both sides at startup, so configuration errors are rapidly discovered
## Configuration (partial)

<table>
<thead>
<tr>
<th>Setting</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Queue Manager</td>
<td>[QM1]</td>
</tr>
<tr>
<td>Bridge Input Queue</td>
<td>[SYSTEM.BLOCKCHAIN.INPUT.QUEUE]APPL1.BLOCKCHAIN.INPUT.QUEUE</td>
</tr>
<tr>
<td>MQ Channel</td>
<td>[]</td>
</tr>
<tr>
<td>MQ Conname</td>
<td>[]</td>
</tr>
<tr>
<td>MQ CCNT URL</td>
<td>[]</td>
</tr>
<tr>
<td>JNDI implementation class</td>
<td>[com.sun.jndi.fscontext.RefFSContextFactory]</td>
</tr>
<tr>
<td>JNDI provider URL</td>
<td>[]</td>
</tr>
<tr>
<td>MQ Userid</td>
<td>[]</td>
</tr>
<tr>
<td>MQ Password</td>
<td>[]</td>
</tr>
</tbody>
</table>

### User Identification

<table>
<thead>
<tr>
<th>Setting</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Userid</td>
<td>[]</td>
</tr>
<tr>
<td>Password</td>
<td>[]</td>
</tr>
<tr>
<td>API path for login</td>
<td>[auth/users/login]</td>
</tr>
</tbody>
</table>

### REST Server

<table>
<thead>
<tr>
<th>Setting</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Address for Composer REST server</td>
<td>[]localhost:3000</td>
</tr>
</tbody>
</table>

### Certificate stores for TLS connections

<table>
<thead>
<tr>
<th>Setting</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Personal keystore</td>
<td>[]</td>
</tr>
</tbody>
</table>
Composer REST API

GET /Trader
Find all instances of the model matched by filter from the data source.

Response Class (Status 200)
Request was successful

Model Example Value

```
{
    "$class": "org.acme.biznet.Trader",
    "tradeId": "string",
    "firstName": "string",
    "lastName": "string"
}
```

POST /Trader
Create a new instance of the model and persist it into the data source.

PUT /Trader/{id}
Find a model instance by {id} from the data source.
IBM MQ – Bridges

Application interface

• Applications interact with the bridge by putting a JSON-formatted message on the queue
  – Sample app provided to show construction of fields

```json
{
  "method": "POST",
  "path": "api/Trader",
  "body": {
    "$class": "Trader",
    "tradeId": "Trader2",
    "firstName": "Jane",
    "lastName": "Doe"
  }
}
```

• Response contains requested data and indication of success or failure)
  – Errors may come directly from the bridge (eg badly formatted input message), from Hyperledger fabric (eg unknown method), or from the chaincode itself (eg bad function parameters)

```json
{ "statusCode": 200,
  "statusType": "SUCCESS",
  "data": [{
    "$class": "Trader",
    "firstName": "John",
    "lastName": "Doe",
    "tradeId": "Trader1"
  }]
}
```

```json
{ "statusCode": 500,
  "statusType": "FAILURE",
  "message": "Error trying invoke business network. Error: chaincode error (status: 500, message: Error: Failed to add object with ID 'Trader1' as the object already exists)"
}
```
IBM MQ
BRIDGE TO SALESFORCE
IBM and Salesforce partnership

IBM and Salesforce announce a new landmark partnership

Read the press release

Accelerate decision making and drive greater customer success

IBM and Salesforce will deliver joint solutions designed to leverage artificial intelligence and enable companies to make smarter decisions, faster than ever before. The partnership will bring insights from Watson directly into the Salesforce Intelligent Customer Success Platform, combining deep customer insights from Salesforce Einstein with Watson’s structured and unstructured data across many industries including healthcare, financial services, retail and weather.

Sign up for our webinar to learn how you can bring IBM and Salesforce to your organization.

Register for webinar
Safesforce’s cloud-based CRM platform enables events to be emitted when changes are made to data, or when applications run. It allows apps to consume these events.

- You can inject these Salesforce events into your MQ systems using the new MQ Bridge for Salesforce with no need for your backend applications to connect to Salesforce
- You can create events in your MQ applications that are published to applications running in Salesforce

Supports Salesforce Platform Events and PushTopics

Uses MQ's publish/subscribe capabilities

The bridge runs on xLinux, but connects to any queue manager and is enabled for monitoring with system topic metrics.
IBM MQ to Salesforce bridge

initial availability in MQ 9.0.2
updated for MQ 9.0.4
IBM MQ – Bridges

Design

- Salesforce Platform
  - Platform Event
  - Push Topic
- Cometd
  - Channel
- Event Store
- MQ Bridge
  - Bayeux
- Queue Manager
  - Topic
  - MQ Client
- MQ App
- Monitoring

SUB
IBM MQ – Bridges

Design

- IBM MQ Bridge
- Queue Manager
- Monitoring
- Salesforce Platform
- Platform Event
  - Push Topic
- Cometd
  - Channel
- Event Store
- REST API
- MQ Client
- Topic
- MQ App
- Monitoring
- PUB
- SUB
What are PushTopics and Platform Events

- PushTopics are "implicit" events generated by changes to Salesforce database tables
- Caused by a SOQL (SQL-ish) statement
- Whenever a table is changed, the SOQL statement is executed to see if it is satisfied
  - If so, PushTopic is created containing requested fields

- Platform Events are "explicit" events, created by user-written application code
- New capability in Salesforce since March 2017
- Administrator defines the name of the event, and the fields it should contain

- Both are made available via a pub/sub API and protocol to any application
TABLE: "CASE"

<table>
<thead>
<tr>
<th>ID</th>
<th>NAME</th>
<th>STATUS</th>
<th>SUBJECT</th>
</tr>
</thead>
<tbody>
<tr>
<td>AAAAAA</td>
<td>Customer1</td>
<td>Normal</td>
<td>Paper jam</td>
</tr>
</tbody>
</table>
### TABLE: "CASE"

<table>
<thead>
<tr>
<th>ID</th>
<th>NAME</th>
<th>STATUS</th>
<th>SUBJECT</th>
</tr>
</thead>
<tbody>
<tr>
<td>AAAAAA</td>
<td>Customer1</td>
<td>Normal</td>
<td>Paper jam</td>
</tr>
<tr>
<td>AAABBB</td>
<td>Customer2</td>
<td>Normal</td>
<td>Phone won't stop ringing</td>
</tr>
</tbody>
</table>
# TABLE: "CASE"

<table>
<thead>
<tr>
<th>ID</th>
<th>NAME</th>
<th>STATUS</th>
<th>SUBJECT</th>
</tr>
</thead>
<tbody>
<tr>
<td>AAAAAA</td>
<td>Customer1</td>
<td>Normal</td>
<td>Paper jam</td>
</tr>
<tr>
<td>AAABBB</td>
<td>Customer2</td>
<td>Normal</td>
<td>Phone won't stop ringing</td>
</tr>
<tr>
<td>BBCC</td>
<td>Customer3</td>
<td>Escalated</td>
<td>Printer on fire</td>
</tr>
</tbody>
</table>

**GENERATE PUSHTOPIC**
Defining a PushTopic

- Can be done by executing APEX code
  - Others APIs exist

```java
PushTopic pushTopic = new PushTopic();
pushTopic.Name = 'EscalatedCaseUpdates';
pushTopic.Query = 'SELECT Id, AccountId, Subject FROM CASE
    WHERE Status = \'Escalated\' AND AccountId != NULL';
pushTopic.ApiVersion = 39.0;
pushTopic.NotifyForOperationCreate = true;
pushTopic.NotifyForOperationUpdate = true;
pushTopic.NotifyForOperationUndelete = true;
pushTopic.NotifyForOperationDelete = true;
pushTopic.NotifyForFields = 'Referenced';
insert pushTopic;
```

- PushTopic will now be generated automatically whenever the Query is satisfied
Defining a Platform Event

Platform Event Definition Edit

Platform Event Information

Deployment Status

IBM MQ – Bridges
Platform Event definition in Salesforce

Standard Fields

- **Action**: Created By, Created Date, Replay ID
- **Field Label**: Created By, Created Date, Replay ID
- **Field Name**: CreatedBy, CreatedDate, ReplayId
- **Data Type**: Lookup(User), Date/Time, External Lookup

Custom Fields & Relationships

- **Action**: Edit, Del
- **Field Label**: MyText, Name
- **API Name**: MyText__c, Name__c
- **Data Type**: Text(64), Text(32)
Creating a Platform Event

- Can be done by executing APEX code
  - Others APIs exist

```java
MQPlatformEvent1__e pe;

pe = new MQPlatformEvent1__e(Name__c='Mark');

Database.SaveResult result = EventBus.publish(pe);

if (result.isSuccess()) {
    System.debug('Published OK');
} else {
    for (Database.Error err : result.getErrors()) {
        System.debug('Failed with ' + err.getStatusCode() + err.getMessage());
    }
}
```
Getting started with the bridge

• Configure the bridge
  – Connection details for MQ (queue manager name, channel, userid etc)
  – Connection details for Salesforce (id, password, OAuth secret, URL etc)
  – TLS keystores (uses same stores for both sides of bridge)
  – Topic information (which topics to listen to from Salesforce, publication root for MQ, QoS)

• Configure Salesforce
  – Define pushtopic rules, define platform event names, add any event generation code

• Configure MQ
  – Define queue for synchronisation values
  – Define topic root, set security rules

• And then start the bridge so that any MQ application can send and receive publications
MQ Applications - Subscribing

• Any MQ application can subscribe to the topics
  – Standard MQ APIs for accessing these
• "Application" could be IIB message flow that exploits Salesforce node for further processing

• Data is JSON format (option to republish full data or just sobject/payload)

```json
{
  "clientId":"w211mr7nytfdadqzulpq4sbovyh",
  "data": {
    "event": {
      "createdDate":"2016-11-12T19:09:31.384Z", "replayId":10, "type":"created"
    },
    "sobject": {
      "AccountId":"Y000009Yxb", "Id":"Y000001QGp", "Subject":"Acct updated"
    }
  },
  "channel": "/topic/EscalatedCaseUpdates"
}
```
IBM MQ – Bridges

Qualities of Service

• The bridge supports two QoS
  – At most once delivery
  – At least once delivery

• All events from Salesforce have a sequence number (replayId) unique to the event name
  – Events are held inside Salesforce for potential replay for up to 24 hours
  – The sequence number of last event processed is persisted in MQ
  – On restart, bridge can use that sequence number to request all events since the last one processed
    • At least once QoS
    – Or can just ask for only new events
      • At most once QoS

• If the bridge out of contact with Salesforce for more than 24 hours, old events are not available
  – Even with at least once QoS
IBM MQ – Bridges

MQ Applications - Publishing

• As well as subscribing to Salesforce topics, bridge subscribes to MQ topics
• MQ applications publish to that tree
• Bridge sends them to Salesforce
  – Then available to any Salesforce application
  – Including any MQ apps listening via the bridge

• Publications contain JSON corresponding to Platform Event fields
  – MQ topic name indicates Salesforce event name

```
Topic:
  /<root>/mqtosfb/event/MQPlatformEvent1__e

Message Body:
  {“MyText__c” : “Some text here”, “Name__c” : “Bob Smith” }
```
IBM MQ – Bridges

Monitoring using the MQ Console

IBM MQ Console

sf1 ▼ Tab 2 ▼ +

Add widget

**Total Events**

**Total PushTopics**

10:34:23

QM1 10

**Demo topic (DEMOPT1)**

**Total Events**

10:34:23

QM1
IBM MQ – Bridges

Monitoring using Grafana
POSSIBLE ENHANCEMENTS
IBM MQ – Bridges

Possible enhancements

• RFEs have been raised that would, if implemented, lead to features such as

• Multiple bridges on a queue manager

• Different QoS for inbound (SF to MQ) and outbound (MQ to SF) publications

• Workload sharing for outbound work
IBM MQ – Bridges

Demo Outline

- A change is made to a Salesforce table
- Resulting in a pushtopic being generated
- Picked up by the bridge
- And republished to an MQ application that is subscribed to the topic

Round trip publication
DEMO
For more info on Salesforce event processing

- MQ Knowledge Centre

- MQ blog entry (and link to video) at

- PushTopics described at

- Platform Events at
  - https://developer.salesforce.com/docs/atlas.en-us.platform_events.meta/platform_events/platform_events_intro.htm

- Testing with Salesforce – Can try these out using FREE developer accounts
KAFKA CONNECTORS
Apache Kafka is an Open-Source Streaming Platform

Use cases
- Messaging
- Website activity tracking
- Metrics
- Log aggregation
- Stream processing
- Event sourcing
- Commit log
IBM MQ – Bridges

Kafka Connect

Over 80 connectors
- HDFS
- Elasticsearch
- MySQL
- JDBC
- IBM MQ
- MQTT
- CoAP
- + many others

https://www.confluent.io/product/connectors/
IBM MQ – Bridges

Kafka Connect MQ Source connector

Open-source – build it yourself
Use any supported MQ release
Uses JMS client internally
Client connections
Supports TLS, authentication
MQ queue -> Kafka topic
Support for binary, text, JSON
Easy to extend

https://github.com/ibm-messaging/kafka-connect-mq-source
IBM MQ – Bridges

Kafka Connect MQ Sink connector

Open-source – build it yourself
Use any supported MQ release
Uses JMS client internally
Client connections
Supports TLS, authentication
Kafka topic -> MQ queue
Support for binary, text, JSON
Easy to extend

https://github.com/ibm-messaging/kafka-connect-mq-sink
Summary

- Simple extension allows MQ apps to exploit new services
  - Event-driven Salesforce applications
  - Integration with Hyperledger Fabric Blockchain
  - Integration with Kafka event streaming

- Follows existing MQ application models

- Easy configuration

- Easy to extend existing processes
Any questions?