BIG Connectivity with WebSphere MQ and WebSphere Message Broker [z/OS & Distributed]

Chris J Andrews and Dave Gorman
IBM

Tuesday February 5th 2013
Session Number 12626
Agenda

- MQ
  - WebSphere MQ Extended Reach (MQXR)
  - WebSphere MQ HTTP Bridge
- Message Broker
  - Introduction to Worklight
  - Worklight Adapters
  - Message Broker Mobile Patterns
    - Mobile enablement for Microsoft .NET applications
    - Create flexible mobile services on top of Message Broker
    - Outbound push notifications for asynchronous data delivery
    - Resource handler including security and caching
Tightly Coupled Systems

Digital devices have now been embedded into systems for over 40 years.

Typically they have used propriety interfaces, tightly coupling the devices to their data capture systems.
Embedded and Mobile Devices

The proliferation of devices has risen dramatically in recent times:

- Popularisation of custom embedded circuitry
- Mobile Phones / Tablets
- Appreciation by industry as to the possibilities of making data available to the user
Billions of smart devices instrument our world today
HTTP serves as the de-facto protocol for communication between browsers and the internet.

What protocol should machines use to communicate with each other?

A common Machine to Machine (M2M) protocol

Internet of Things
MQ Telemetry Transport (MQTT)

To save inventing a new protocol every time a new embedded device came along, a common protocol is needed.

MQTT is that protocol. It traces its roots back to 1999, where Dr Andy Stanford-Clark of IBM, and Arlen Nipper of Arcom (now Eurotech) devised the protocol.

Design goals of MQTT:

- Works over unreliable communication networks
- Minimal data overhead (low bandwidth)
- Capable of supporting large numbers of devices
- Simple to interface the data with the traditional IT world
- Simple to developers to write applications to use
MQ Telemetry Transport (MQTT)

- Expect and cater for frequent network disruption – built for low bandwidth, high latency, unreliable, high cost networks

- Expect that client applications may have very limited resources available.

- Publish/subscribe messaging paradigm as required by the majority of SCADA and sensor applications.

- Provide traditional messaging qualities of service where the environment allows.

- Published protocol for ease of adoption by device vendors and third-party client software.
MQTT Sample Usage Applications

Medical devices in hospital equipment

Facebook Messenger

Low latency (milliseconds)
Low battery usage
Uses data sparingly
Implemented within weeks

The Andy Stanford-Clark Mouse Trap State Advisor
MQTT Qualities of Service

**QoS 0: At most once delivery (non-persistent)**
- No retry semantics are defined in the protocol.
- The message arrives either once or not at all.

**QoS 1: At least once delivery (persistent, dups possible)**
- Client sends message with Message ID in the message header
- Server acknowledges with a PUBACK control message
- Message resent with a DUP bit set if the PUBACK message is not seen

**QoS 2: Exactly once delivery (persistent)**
- Uses additional flows to ensure that message is not duplicated
- Server acknowledges with a PUBREC control message
- Client releases message with a PUBREL control message
- Server acknowledges completion with a PUBCOMP control message
MQTT Power Usage

How does MQTT use power?
- HTC Android mobile phone

<table>
<thead>
<tr>
<th>Keep Alive (Seconds)</th>
<th>% Battery / Hour</th>
<th>3G</th>
<th>Wifi</th>
</tr>
</thead>
<tbody>
<tr>
<td>60</td>
<td>0.77641278</td>
<td>0.0119021</td>
<td></td>
</tr>
<tr>
<td>120</td>
<td>0.38884457</td>
<td>0.0062861</td>
<td></td>
</tr>
<tr>
<td>240</td>
<td>0.15568461</td>
<td>0.00283991</td>
<td></td>
</tr>
<tr>
<td>480</td>
<td>0.07792208</td>
<td>0.00134018</td>
<td></td>
</tr>
</tbody>
</table>

Protocol allows tuning to suit devices
MQTT Data Usage

How does MQTT compare to HTTP for data usage?

<table>
<thead>
<tr>
<th>Scenario</th>
<th>HTTP</th>
<th>MQTT n3</th>
</tr>
</thead>
<tbody>
<tr>
<td>1. Getting a single piece of data from the server</td>
<td>126 bytes</td>
<td>69 bytes</td>
</tr>
<tr>
<td>2. Putting a single piece of data to the server</td>
<td>141 bytes</td>
<td>47 bytes</td>
</tr>
<tr>
<td>3. Getting 100 pieces of data from the server</td>
<td>12600 bytes</td>
<td>2445 bytes</td>
</tr>
<tr>
<td>4. Putting 100 pieces of data to the server</td>
<td>14100 bytes</td>
<td>2126 bytes</td>
</tr>
</tbody>
</table>

Very favourably – of the order of a 5x saving!
WebSphere MQ Telemetry

Supplied as a component WebSphere MQ V7.1 and v7.5, under the component name “WebSphere MQ Extended Reach” (or MQXR).

MQXR brings MQTT protocol functionality to WebSphere MQ!

- Highly scaleable: 100,000+ clients
- Security: SSL channels, JAAS authentication, WMQ OAM
- Ships with reference Java and C clients
  - Small footprint clients
  - other APIs and implementations of MQTT available via 3rd parties
WebSphere MQ Telemetry – Further Reading

MQTT homepage:
http://mqtt.org

MQTT Specification

WebSphere MQ and MQ Telemetry
http://www-01.ibm.com/software/integration/wmq/

MQTT: the Smarter Planet Protocol
http://andypiper.co.uk/2010/08/05/mqtt-the-smarter-planet-protocol/

Lotus Expeditor (micro broker)
 Agenda

- MQ
  - WebSphere MQ Extended Reach (MQXR)
  - WebSphere MQ HTTP Bridge
- Message Broker
  - Introduction to Worklight
  - Worklight Adapters
  - Message Broker Mobile Patterns
    - Mobile enablement for Microsoft .NET applications
    - Create flexible mobile services on top of Message Broker
    - Outbound push notifications for asynchronous data delivery
    - Resource handler including security and caching
The WebSphere HTTP Bridge grants HTTP client applications the ability to access WebSphere MQ messages on queues and topics.

The HTTP Bridge comprises of a JEE Web application (servlet), which is to be installed into a JEE Application server in order to be used.
WebSphere MQ HTTP Bridge

The WebSphere MQ HTTP Bridge provides two key benefits:

1) **Zero Client Footprint.**
   No WebSphere MQ MQI client libraries are required on the application host.
   In addition, *any* platform which supports HTTP can access WebSphere MQ data.

2) **Simplifies access to WebSphere MQ messages** from browser based internet applications.
   No WebSphere MQ programming knowledge is required to program the client applications.
WebSphere MQ HTTP Bridge

How does data access work from HTTP?

<table>
<thead>
<tr>
<th>HTTP Request</th>
<th>Result</th>
</tr>
</thead>
<tbody>
<tr>
<td>POST</td>
<td>Puts a message to a queue or topic (MQPUT)</td>
</tr>
<tr>
<td>GET</td>
<td>Browses the first message on the queue (MQGET with browse)</td>
</tr>
<tr>
<td>DELETE</td>
<td>Receives a message from the queue (destructive MQGET), or creates a non-durable subscription from a topic</td>
</tr>
<tr>
<td>PUT</td>
<td>Not used</td>
</tr>
</tbody>
</table>

The HTTP request defines the location and name of the queue or topic access point:

POST /msg/queue/myQueue/ HTTP/1.1
Host: myhost.mydomain
WebSphere MQ HTTP Bridge

Example 1: **MQPUT**
Put a message to a queue, with message body containing a string message:

```
POST /msg/queue/myQueue/ HTTP/1.1
Host: myhost.mydomain
Content-Type: text/plain
x-msg-correlID: 1234567890
Content-Length: 60

Here is my message body that is posted on the queue.
```

This HTTP POST response is of the form:

```
HTTP/1.1 200 OK
Date: Wed, 2 Jan 2007 22:38:34 GMT
Server: Apache-Coyote/1.1 WMQ-HTTP/1.1 JEE-Bridge/1.1
Content-Length: 0
```
WebSphere MQ HTTP Bridge

Example 2: **MQGET**
Destructively receive a message from a queue, waiting a maximum of 10 seconds:

```
DELETE /msg/queue/myQueue/ HTTP/1.1
Host: myhost.mydomain
x-msg-wait: 10
x-msg-require-headers: correlID
```

This HTTP DELETE response is of the form:

```
HTTP/1.1 200 OK
Date: Wed, 2 Jan 2007 22:38:34 GMT
Server: Apache-Coyote/1.1 WMQ-HTTP/1.1 JEE-Bridge/1.1
Content-Length: 60
Content-Type: text/plain; charset=utf-8
x-msg-correlId: 1234567890
```

Here is my message body from the queue.
Agenda

- MQ
  - WebSphere MQ Extended Reach (MQXR)
  - WebSphere MQ HTTP Bridge
- Message Broker
  - Introduction to Worklight
  - Worklight Adapters
  - Message Broker Mobile Patterns
    - Mobile enablement for Microsoft .NET applications
    - Create flexible mobile services on top of Message Broker
    - Outbound push notifications for asynchronous data delivery
    - Resource handler including security and caching
Worklight Overview

**Worklight Studio**
The most complete, extensible environment with maximum code reuse and per-device optimization

**Worklight Server**
Unified notifications, runtime skinning, version management, security, integration and delivery

**Worklight Runtime Components**
Extensive libraries and client APIs that expose and interface with native device functionality

**Worklight Console**
A web-based console for real-time analytics and control of your mobile apps and infrastructure

Complete your sessions evaluation online at SHARE.org/SFEval
Worklight Architecture

1. Worklight Studio
   - HTML5, Hybrid and Native Coding
   - Optimization Framework
   - Integrated Device SDKs
   - 3rd-Party Library Integration

2. Worklight Server
   - Server-side Application Code
   - JSON Translation
   - Authentication
   - Adapter Library

3. Device Runtime
   - Cross-platform Compatibility Layer
   - Server Integration Framework
   - Encrypted Storage
   - Runtime Skinning
   - Reporting for Statistics and Diagnostics

4. Worklight Console
   - Application Version Management
   - Push Management
   - Reporting and Analytics

Complete your sessions evaluation online at SHARE.org/SFEval
Worklight Overview

Worklight Server
- Server-side Application Code
- JSON Translation
- Authentication
- Adapter Library
- Server Aggregation
- Client-side App Resources
- Direct Update
- Mobile Web Apps
- Unified Push Notifications

Device Runtime
- PhoneGap
- Cross Platform Technology
- Worklight
- Security and Authentication
- Back-end Data Integration
- Post-deployment control and Diagnostics

Complete your sessions evaluation online at SHARE.org/SFEval
Types of Mobile Application

**Browser Access**
Written in HTML5 JavaScript and CSS3. Quick and cheap to develop, but less powerful than native.

**Hybrid Apps - Web**
HTML5 code and Worklight runtime libraries packaged within the app and executed in a native shell.

**Hybrid Apps - Mixed**
User augments web code with native language for unique needs and maximized user experience.

**Native Apps**
Platform-specific. Requires unique expertise, pricy and long to develop. Can deliver higher user experience.
Agenda

• MQ
  • WebSphere MQ Extended Reach (MQXR)
  • WebSphere MQ HTTP Bridge
• Message Broker
  • Introduction to Worklight
  • **Worklight Adapters**
  • Message Broker Mobile Patterns
    • Mobile enablement for Microsoft .NET applications
    • Create flexible mobile services on top of Message Broker
    • Outbound push notifications for asynchronous data delivery
    • Resource handler including security and caching
Worklight Adapters

- Adapters provide the glue between Worklight and back-end applications
  - Provides the extensibility mechanism for Worklight to call out to back-end systems
- Worklight has two built-in interfaces that adapters can use (HTTP and SQL)
  - Worklight has client-side JavaScript APIs so that applications can invoke services
  - Likewise, server-side JavaScript APIs are available to implement procedures (adapters)
Worklight Adapters

- An adapter contains two files for configuration and implementation
  - The first file is XML and contains the overall metadata (procedure names, protocol etc)
  - Second file is JavaScript and contains one function (procedure) for each entry point

- Adapters are uploaded to Worklight Server ready for mobile applications
  - Once deployed, adapters are managed through the Worklight Console
Invoking Worklight adapters

- Adapters are invoked from mobile applications using HTTP/JSON
  - This convention makes Worklight adapters easy to test using web browsers
  - Client side applications use the XMLHttpRequest object for asynchronous calls
  - Mobile toolkits (JQuery, Dojo and Sencha) wrap this in a device independent layer
Agenda

- MQ
  - WebSphere MQ Extended Reach (MQXR)
  - WebSphere MQ HTTP Bridge
- Message Broker
  - Introduction to Worklight
  - Worklight Adapters
  - **Message Broker Mobile Patterns**
    - Mobile enablement for Microsoft .NET applications
    - Create flexible mobile services on top of Message Broker
    - Outbound push notifications for asynchronous data delivery
    - Resource handler including security and caching
Worklight to Microsoft .NET Service Enablement

- Creates a mobile-ready service around a Microsoft .NET application
  - Generates a web service implementation which is deployed to Message Broker
  - Builds a Worklight integration adapter and a sample mobile application
  - Inbound data from the mobile application is sent to Worklight as JSON/HTTP
  - The adapter converts the JSON data into/from SOAP/HTTP for the .NET web service
Configuring the Pattern Instance

- Pattern is configured with Microsoft .NET and Worklight information
  - Server address is a key field as it is used to configure both ends of the connection!
  - Standard set of error handling and logging options are provided by the pattern
  - Adapter configured with the maximum number of concurrent (HTTP) connections
  - Once this limit is reached, Worklight will queue inbound requests from applications
Configuring the Microsoft .NET Assembly

- User-defined editor allows the pattern user to select their .NET assembly
  - Selection proceeds to a class and the (static) methods available in that class
  - Assembly can be developed in any .NET language (for example, VB.NET or C#)
  - Return value and parameters are reflected on and displayed by the user-defined editor
Generated Message Broker Projects

- The pattern generates an application and a library
  - Application contains the mechanics of the pattern instance
  - Library contains subflows for user customizations
  - Customizations are never deleted on re-generation!

- WSDL represents the selected .NET methods
  - One WSDL operation for each .NET (static) method
  - Likewise one message part defined per operation
  - WSDL types are defined in a separate XML schema file
  - WSDL and XSD are deployed directly to Message Broker
Worklight Adapter

- Worklight adapter generated which reflects the web service methods
  - Integrates the mobile application with the Message Broker .NET web service
  - One procedure is generated for each operation (method) on the web service
  - Adapter manages the conversion between JSON and SOAP/XML data formats
  - Adapter generated in a separate project so it can be deployed to Worklight Server
Mobile Application

- Pattern also creates a mobile application to test the Worklight adapter
  - Each operation has views (pages) to configure and invoke the back-end service
  - Application is built using Dojo Mobile (ensures it is device independent)
  - More information on the Dojo mobile toolkit here: http://dojotoolkit.org/features/mobile
Mobile Application

- The mobile application has a single mobile web environment
  - Application is best suited for browsers on small screen mobile devices
  - Easy to add extra environments for iOS, Android and many more!

- Android development requires a separate download (Android SDK)
  - Pick and choose your target Android versions from Android SDK Manager
Agenda

- MQ
  - WebSphere MQ Extended Reach (MQXR)
  - WebSphere MQ HTTP Bridge
- Message Broker
  - Introduction to Worklight
  - Worklight Adapters
  - **Message Broker Mobile Patterns**
    - Mobile enablement for Microsoft .NET applications
    - Create flexible mobile services on top of Message Broker
    - Outbound push notifications for asynchronous data delivery
    - Resource handler including security and caching
Worklight Mobile Services

- Creates a mobile-ready interface around a Message Broker service
  - Services are a first class artifact in Message Broker alongside applications and libraries
  - Builds an adapter to integrate Worklight and Message Broker services
  - Inbound data from the mobile application is sent to Worklight as JSON/HTTP
  - Makes it very simple to mobile enable a Message Broker service!
    - The adapter passes the inbound request straight through to the service
    - Pattern adds an HTTP/JSON message flow (binding) to the service project
Configuring the Pattern Instance

- Create a Message Broker service and then instantiate the pattern
  - You choose which operations in the service are available to mobile applications
  - Standard set of Worklight pattern parameters provided to configure the adapter
Configuring the Pattern Instance

- The mobile service pattern can also be launched from the Navigator
  - Intuitive user experience for mobile enablement of Message Broker services
  - The selected service name is passed to the pattern as the launch configuration
  - Pattern instance is configured automatically and can be immediately generated
Worklight Adapter

- Generates a Worklight adapter which reflects the web service methods
  - Integrates the mobile application with the Message Broker web service
  - One procedure is generated for each selected operation in the service
  - Request-response and one-way interactions for the service are supported
Agenda

- MQ
  - WebSphere MQ Extended Reach (MQXR)
  - WebSphere MQ HTTP Bridge
- Message Broker
  - Introduction to Worklight
  - Worklight Adapters
  - **Message Broker Mobile Patterns**
    - Mobile enablement for Microsoft .NET applications
    - Create flexible mobile services on top of Message Broker
    - **Outbound push notifications for asynchronous data delivery**
    - Resource handler including security and caching
Worklight Push Notification Services

- Worklight supports asynchronous push notifications to mobile applications
  - Push notifications have a measurable impact on the success of mobile applications
  - There are many IT challenges in supporting push notifications (devices, delivery etc)
- Push notifications are applicable across many industry verticals
  - Healthcare, retail, travel, transportation, government, insurance and more!
- All the major mobile platforms support push notification services
  - Apple iOS 3, Google Android 2.2, RIM Blackberry 5 and Windows Phone 7

Complete your sessions evaluation online at SHARE.org/SFEval
Worklight Push Notification Services

- Users receive notifications when the mobile application is not active
  - Efficiency gain as application does not need to issue constant queries
  - Saves battery life and also reduces network bandwidth (communication fees)
- Notifications are not always appropriate and have disadvantages
  - Users need to subscribe on their device to receive push notifications
  - Notifications are limited in the size of their payload (for example, 256 bytes on iOS)
  - No quality of service is guaranteed and there is no delivery notification
  - No guarantee either that the end-to-end delivery chain is secure
Worklight Push Notification from WebSphere MQ

- Creates a push notification adapter from a WebSphere MQ queue
  - Generates a web service implementation which is deployed to Message Broker
  - Builds a Worklight integration adapter which polls for pending notifications
  - Pending notifications are written to a WebSphere MQ queue by a provider application
  - The adapter converts the notifications into JSON and arranges delivery to the mobile

[Diagram of the process]
Configuring the Pattern Instance

- Pattern is configured with Worklight and Message Broker information
  - Server address is a key field as it is used to configure both ends of the connection!
  - Standard set of error handling and logging options are provided by the pattern
  - Application specific fields can be delivered in the push notification
  - Configured as part of the pattern instance so that an accurate schema can be created
Worklight Adapter

- Worklight adapter generated which periodically checks for notifications
  - Integrates Worklight with a queue of notifications managed by Message Broker
  - Generated pattern instance project includes a schema for the notification messages
  - Adapter manages the conversion from XML to JSON for the Worklight server-side calls
- Polling interval for pending notifications is configurable in the pattern
  - Adapter greedily processes all pending notifications each time it wakes up
Agenda

- MQ
  - WebSphere MQ Extended Reach (MQXR)
  - WebSphere MQ HTTP Bridge
- Message Broker
  - Introduction to Worklight
  - Worklight Adapters
  - Message Broker Mobile Patterns
    - Mobile enablement for Microsoft .NET applications
    - Create flexible mobile services on top of Message Broker
    - Outbound push notifications for asynchronous data delivery
    - Resource handler including security and caching
Resource oriented architecture is a well known implementation pattern
- Provides a common set of functions (CRUD – Create Read Update and Delete)
- This pattern provides an adapter which implements CRUD operations
- A Message Broker service is generated with subflows for each operation
- The service integrates security authorization and authentication (LDAP)
- Operations optionally integrate with the Message Broker Global Cache (WXS)
Implementing Resource Handlers

- Complete the pattern instance by implementing the resource handlers
  - Subflows are generated for each CRUD operation in a customization project
  - Pattern generates a reference implementation of a back end system in ESQL

- Message Broker has excellent support for enterprise applications
  - Common design pattern to integrate with SAP, Siebel, JDEdwards and PeopleSoft
  - Wizards makes it easy to discover the application content (for example, SAP iDocs)
  - Rich SAP support includes iDocs, ALE, BAPI and query SAP tables (QISS)
WebSphere Extreme Scale (WXS)

- WebSphere Extreme Scale is tightly integrated with Message Broker
  - Provides a highly scalable, fault tolerant, elastic in-memory data grid
  - One or more execution groups manage a single logical cache of key-value data
  - WXS components are hosted within the execution group processes
  - Default scope is one cache per broker but this can be extended to multiple brokers
  - Vital for mobile applications where the number of devices can be huge
  - Caching fits perfectly with a CRUD model of many readers and (generally) few writers
  - Message Broker activity log shows the cache activity as CRUD operations complete

![Message Broker activity log](image)
Authorization and Authentication

- Patterns provides a security model based around LDAP
  - Caching fits perfectly with a CRUD model of many readers and (generally) few writers
  - Users are authenticated using HTTP basic authentication by the HTTP Input node
  - Authorization is then done by splitting the users into two groups (readers/writers)
  - A user is authorized if they are a member of the group in the LDAP directory
  - The LDAP queries are issued by the message flow using the Security PEP node
  - Caching changes are made through WXS after the user has cleared security
<table>
<thead>
<tr>
<th>Time</th>
<th>Monday</th>
<th>Tuesday</th>
<th>Wednesday</th>
<th>Thursday</th>
<th>Friday</th>
</tr>
</thead>
<tbody>
<tr>
<td>08:00</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Are you running too many queue managers or brokers?</td>
</tr>
<tr>
<td>09:30</td>
<td></td>
<td>What's New in WebSphere Message Broker</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>11:00</td>
<td></td>
<td>Extending IBM WebSphere MQ and WebSphere Message Broker to the Cloud</td>
<td>WMQ - Introduction to Dump Reading and SMF Analysis - Hands-on Lab</td>
<td>BIG Data Sharing with the cloud - WebSphere eXtreme Scale and WebSphere Message Broker integration</td>
<td>Getting the best availability from MQ on z/OS by using Shared Queues</td>
</tr>
<tr>
<td>12:15</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>01:30</td>
<td>Introduction to MQ</td>
<td>MQ on z/OS – Vivisection</td>
<td>Migration and maintenance, the necessary evil</td>
<td>The Dark Side of Monitoring MQ - SMF 115 and 116 Record Reading and Interpretation</td>
<td></td>
</tr>
<tr>
<td>03:00</td>
<td>First Steps With WebSphere Message Broker: Application Integration for the Messy</td>
<td>BIG Connectivity with WebSphere MQ and WebSphere Message Broker</td>
<td>WebSphere MQ CHINIT Internals</td>
<td>Using IBM WebSphere Application Server and IBM WebSphere MQ Together</td>
<td></td>
</tr>
<tr>
<td>04:30</td>
<td>WebSphere MQ application design, the good, the bad and the ugly</td>
<td>What’s New in the WebSphere MQ Product Family</td>
<td>MQ &amp; DB2 – MQ Verbs in DB2 &amp; Q-Replication</td>
<td>WebSphere MQ Channel Authentication Records</td>
<td></td>
</tr>
<tr>
<td>06:00</td>
<td></td>
<td>Clustering - The Easier Way to Connect Your Queue Managers</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>