# *MQ Service Provider for z/OS Connect (And some other new stuff)*

**Matt Leming – lemingma@uk.ibm.com**

**MQ Development**

# 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.

# Agenda

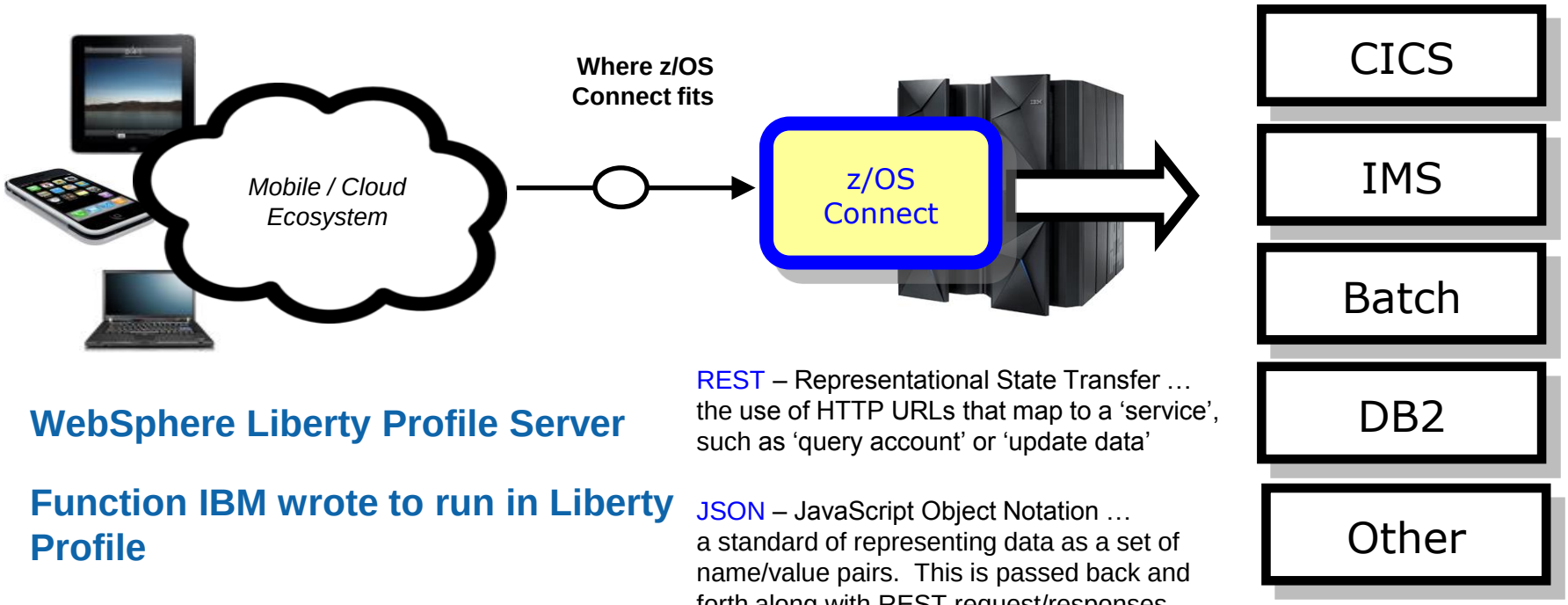
- **MQ Service Provider for z/OS Connect**
  - What is z/OS Connect?
  - What is the MQ Service Provider?
  - What might it look like?

- **Other new stuff**
  - JMS enhancements
  - Enhancements to sub-system connectivity

# What is z/OS Connect?

# z/OS Connect – what is it?

It's about getting REST and JSON into your mainframe environment in a way that enables you to best take advantage of the assets that exist there:




- **WebSphere Liberty Profile Server**
- **Function IBM wrote to run in Liberty Profile**

REST – Representational State Transfer … the use of HTTP URLs that map to a 'service', such as 'query account' or 'update data'

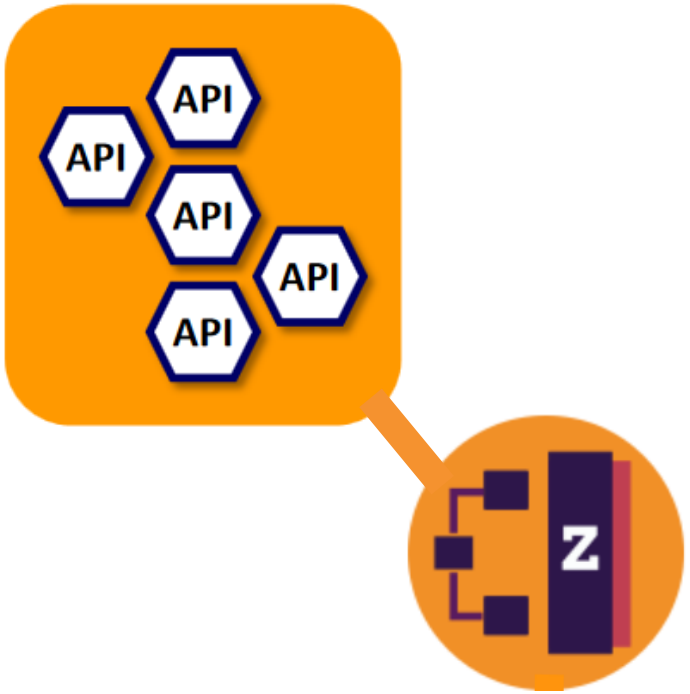
JSON – JavaScript Object Notation … a standard of representing data as a set of name/value pairs. This is passed back and forth along with REST request/responses

# Two versions!

- **Original free version: z/OS Connect V1**
  - Available since 2014

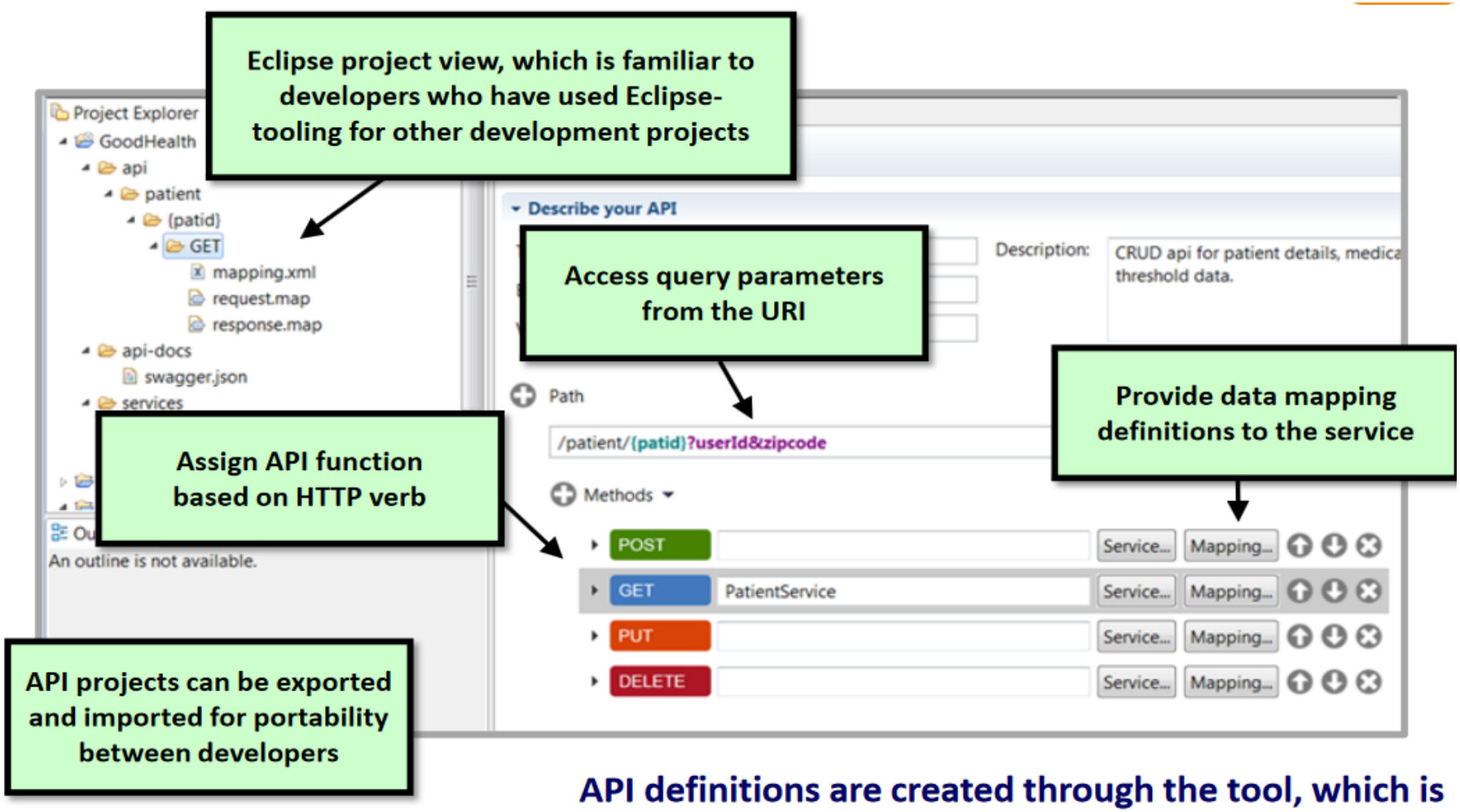
- **Provides**
  - Basic REST/JSON support
  - A basic REST API for discovering and managing exposed services
  - Capable of performing data conversion from JSON to the data format required by the backend configured service
  - Optional authorization checking using SAF to allow or deny users access to z/OS Connect services
  - Optional activity recording using SMF to track requests by date and time, bytes sent and received, and response time

# Two versions!




- **z/OS Connect EE V2**
  - Available since December 2015
  - Chargeable
  - Continuous delivery approach to enhancing product
  - Based on V1 with enhancements…
- **Enhancements over V1**
  - More advanced REST functionality
    - Proper use of HTTP verb and URL to define API
  - API Editor
    - Define APIs, described using Swagger 2
    - Map API down onto underlying service
      - body, header and query parameters
      - pass-through, redaction and default values
    - Deploy APIs plus artefacts to runtime using single file (AAR – Api ARchive)
    - Consistent tooling and API definition for backend systems: CICS, IMS etc

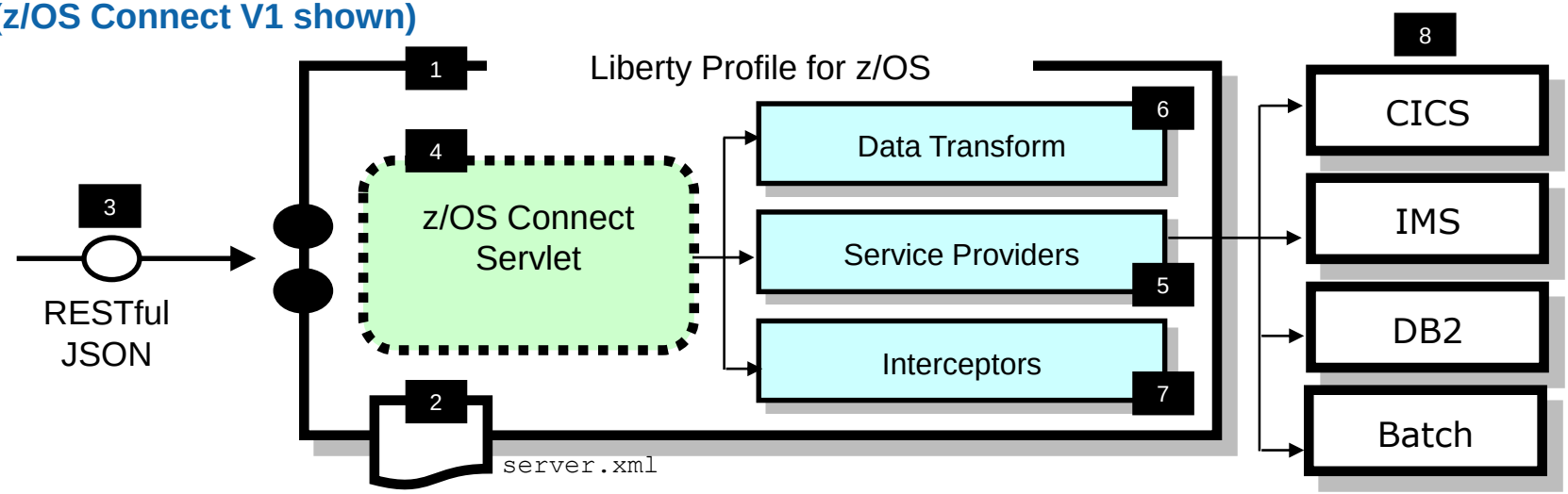
# z/OS Connect EE tooling




**API definitions are created through the tool, which is consistent across backend systems (CICS, IMS, etc.)**

# Summary in one picture

**(z/OS Connect V1 shown)**




1 z/OS Connect is software function that runs in Liberty Profile for z/OS

2 z/OS Connect is described and configured in the Liberty `server.xml` file

3 z/OS Connect is designed to accept RESTful URIs with JSON data payloads

4 One part of z/OS Connect is a servlet that runs in Liberty Profile z/OS.

5 A 'Service Provider' is software that provides the connectivity to the backend system

6 z/OS Connect provides the ability to transform JSON to the layout required by backend

7 'Interceptors' are callout points where software can be invoked to do things such as SAF authorization and SMF activity recording

8 Backend systems supported are CICS, IMS, Batch, and DB2

# What is the MQ Service Provider?

# Statement of general direction

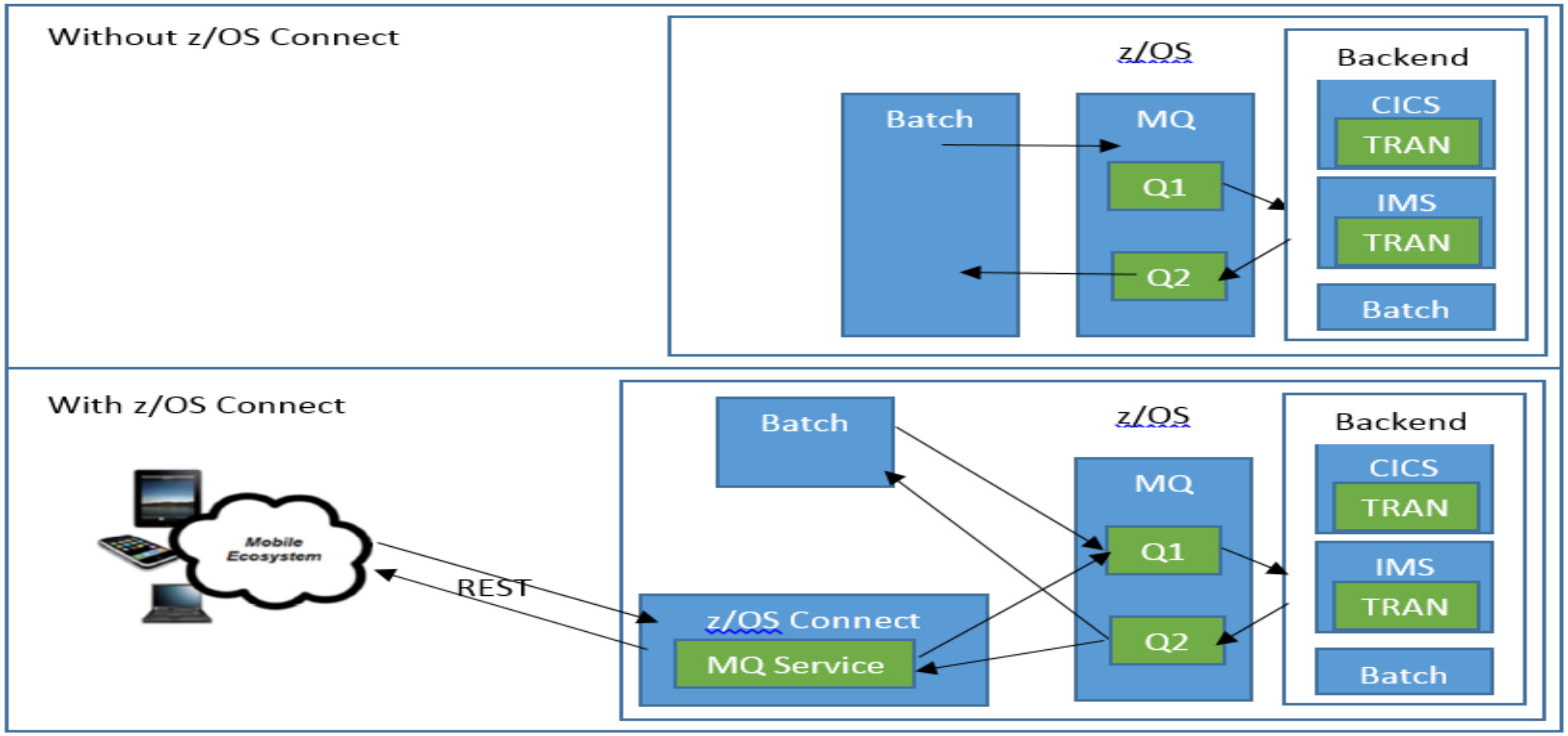
- IBM intends that a future release of IBM MQ for z/OS will provide support for both z/OS Connect and z/OS Connect EE

- https://ibm.biz/MQzOSConnectSOD

# The MQ Service Provider

- **Free of charge z/OS Connect service provider that allows existing services that are fronted by MQ to be accessed via a RESTful front end**
  - Both V1 and EE supported
  - Same capabilities in both versions

- **Clients need have no knowledge of MQ**



# Service types

- **Each URL in z/OS Connect maps to a service**
- **With the MQ Service Provider there are two different types of service**
  - Two way services
  - One way services
- **A two way service provides request/reply messaging:**
  1. Client issues HTTP POST with some payload (JSON)
  2. MQ Service Provider sends payload (optional transformation) to one MQ queue
  3. Back end application processes payload and puts response on reply queue
  4. MQ Service Provider gets response (optional transformation) and sends it to client as the body of the HTTP POST response
- **A one way service exposes standard MQ verbs against a single destination**
  - HTTP POST == MQPUT (queue and topic)
  - HTTP DELETE == MQGET (queue)
  - HTTP GET == MQGET (browse) (queue)
- **Allows more advanced interactions with MQ**

# Two way example

server.xml

```
<zosConnectService id="zosconnMQ40"
                   invokeURI="/mq40"
                   serviceName="mq40"
                   serviceRef="mq40" />

<mqzOSConnectService id="mq40"
                     connectionFactory="jms/cf1"
                     destination="jms/twoWayRequestQ"
                     waitInterval="10000"
                     replyDestination="jms/twoWayResponseQ"/>
```

Provided by z/OS Connect

Provided by MQ Service Provider

- **HTTP POST to https://<hostname>:<port>/mq40**

- **All MQ related information is held in *mqzOSConnectService element***
  - Sensible defaults
  - Overridable via HTTP headers, e.g. *ibm-mq-gmo-waitInterval*
  - Builds on the MQ messaging provider in Liberty. Uses JMS

# Example two-way service

- Insurance quote service which takes a request from an MQ queue and sends a response to another queue
- COBOL request

```
****************************** Top of Da
       01  QUOTREQT.
           05  NAME          PIC A(20).
           05  ADDR          PIC X(20).
           05  ZIPCODE       PIC 9(9).
           05  BIRTHDAY      PIC 9(8) COMP-3.

****************************** Bottom of
```

- COBOL response

```
******************************
       01  QUOTERESP.
           05  ACCEPTED    PIC A(1).
           05  REASON      PIC X(20).
           05  COST        PIC 9(8).

****************************** B
```

- So COBOL -> JSON conversion needed using built in tooling

# Example two-way service - configuration

```
<zosConnectService id="zosconnIQ"
                   invokeURI="/iq"
                   serviceName="iq"
                   serviceRef="iq"
                   dataXformRef="xformJSON2Byte" />

<mqzOSConnectService id="iq"
                     connectionFactory="jms/cf1"
                     destination="jms/iqRequestQ"
                     waitInterval="10000"
                     replySelection="msgIDToCorrelID"
                     replyDestination="iqResponseQ"
                     receiveTextCCSID="37" />

<zosConnectDataXform id="xformJSON2Byte"
                     bindFileLoc="${XFORM_ROOT}/bindfiles"
                     bindFileSuffix=".bnd"
                     requestSchemaLoc="${XFORM_ROOT}/json"
                     requestSchemaSuffix=".json"
                     responseSchemaLoc="${XFORM_ROOT}/json"
                     responseSchemaSuffix=".json" />
```

# Example two-way service – request

POST ▾ https://winmvs41.hursley.ibm.com:12362/iq Send

+ Headers

− Data

Select one of the options below to include data with the request.

Custom ▾

Enter the data and its corresponding MIME type below.

application/json

```
{
      "OLACB01Operation":{
            "quotreqt":{
                  "name": "Matt Leming",
                  "addr": "IBM US",
                  "zipcode": 123456789,
                  "birthday": 10271979
            }
      }
}
```

```
********************************** Top of Da
       01 QUOTREQT.
          05 NAME          PIC A(20).
          05 ADDR          PIC X(20).
          05 ZIPCODE       PIC 9(9).
          05 BIRTHDAY      PIC 9(8) COMP-3.

********************************** Bottom of
```

# Example two-way service – request on queue

| | | | | | |
|---|---|---|---|---|---|
| iqRequest | Local | Queue man... | 0 | 0 | 1 |
| iqResponse | Local | Queue man... | 1 | 0 | 0 |

MQ Service Provider waiting for response

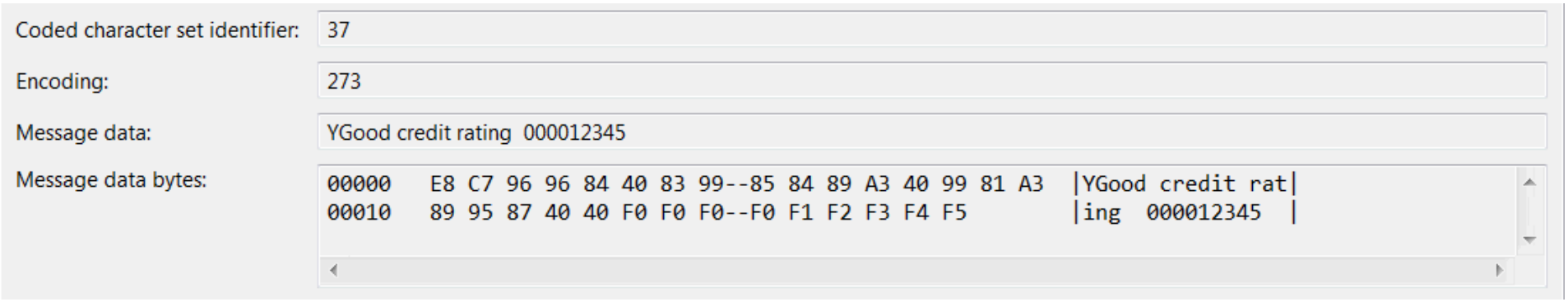
Message on queue

| | |
|---|---|
| Coded character set identifier: | 37 |
| Encoding: | 273 |
| Message data: | Matt Leming IBM US 123456789 Ép¤ |

Message data bytes:

```
00000   D4 81 A3 A3 40 D3 85 94--89 95 87 40 40 40 40 40  |Matt Leming     |
00010   40 40 40 40 C9 C2 D4 40--E4 E2 40 40 40 40 40 40  |    IBM US      |
00020   40 40 40 40 40 40 40 40--F1 F2 F3 F4 F5 F6 F7 F8  |        12345678|
00030   F9 01 02 71 97 9F      --                        |9..Ép¤          |
```

# Example two-way service – response on queue



# Example two-way service – response

Headers | Raw | Preview


```
{"OLACB01OperationResponse":{"quoteresp":{"accepted":"Y","reason":"Good credit
rating","cost":1234}}}
```


```
*****************************************

     01  QUOTERESP.
         05 ACCEPTED   PIC A(1).
         05 REASON     PIC X(20).
         05 COST       PIC 9(8).

*****************************************
```

# Delivery plans

- **MQ Service Provider for z/OS Connect**
  - Will be provided in USS component of a future version of MQ
  - Will be supported with MQ 8 and later queue managers
- **Will also be made available on Fix Central**
  - In case you don't have the latest version of MQ available
- **Only contains the MQ Service Provider function. You need to provide z/OS Connect (V1 or V2)**
- **Code consists of a jar file, and a Liberty feature manifest**

# Beta version available now

- If you want to join the MQ beta then please let me know

# JMS Enhancements

# What is JMS and why do I want it?

- **Java Message Service**
- **Standard API for interacting with messaging middleware such as MQ**
- **Simple support for interacting with both queues and topics**
- **Widely adopted and used throughout the industry**

- **Most recent version is JMS 2**
  - Released in 2013
  - Mainly focused on ease of use (new JMSContext API)
  - Backwards compatible
  - Some minor functionality added
    - Delivery delay being the most notable

# What Java support was there for MQ on z until recently?

- **JMS 1.* supported since MQ v5**
- **JMS 2.0 support added in MQ v8**
- **Supported on z/OS in**
  - z/WAS, including Liberty
  - Batch
- **In CICS**
  - MQ base classes for Java supported in OSGi JVM Server
    - MQ specific Java API
- **In IMS**
  - No Java support with MQ at all

# What has been added?

- **Support for MQ JMS in CICS 5.3 and 5.2 OSGi environments**
  - Works with MQ 7.1 and later
  - Depending on environment some APARs needed
  - Since 1Q 2015
- **Support for MQ JMS in IMS 13**
  - Works with MQ 8 and later
  - Requires MQ APARs if using MQ 8
  - Since 4Q 2015
- **Support for MQ JMS in CICS 5.3 Liberty environments**

# What is CICS Liberty JVM Server?

- **For a number of CICS release support has been provided for running Liberty inside the CICS address space**
  - Only a subset of JEE capability supported
  - Integration provided with native CICS capabilities
  - In some cases slight adjustments to programming model/the way things work

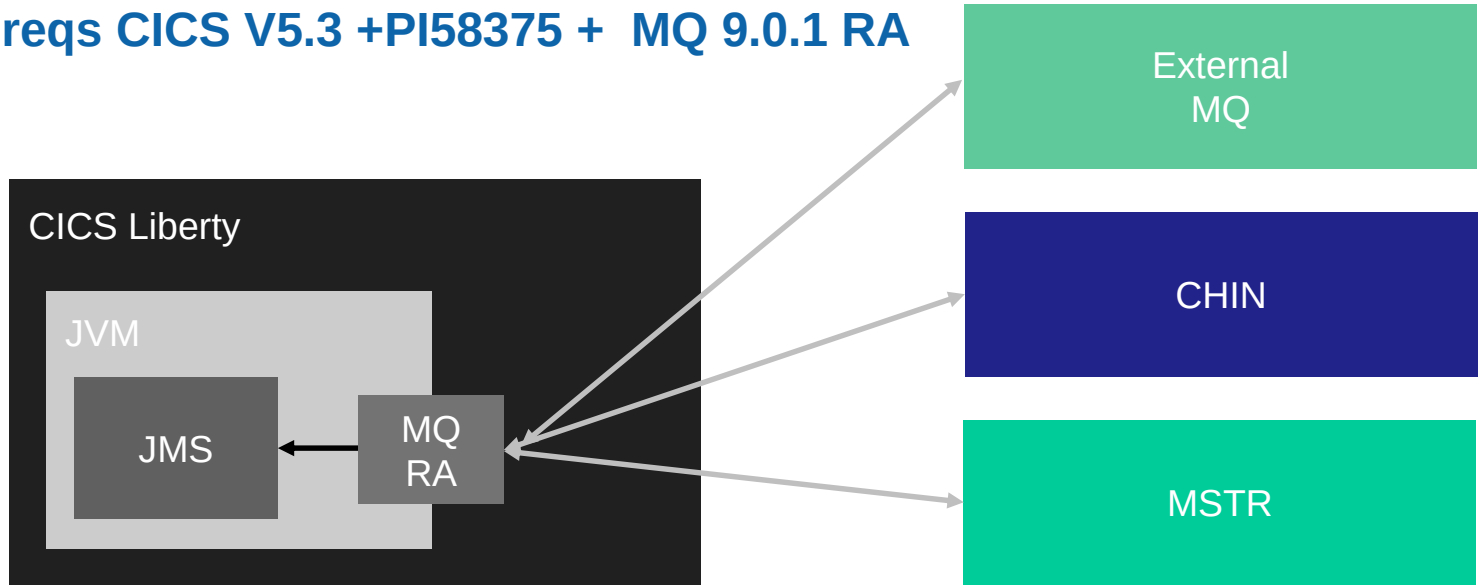
- **CICS V5.3 +PI58375 changes things, two sorts of CICS Liberty introduced**
  - Liberty Standard
  - Liberty Integrated

- **Liberty Standard: FULL Liberty support. No CICS integration**
  - Perfect for ‘lift and shift’

- **Liberty Integrated: Same capability as provided before, subset of JEE integrated with CICS capabilities**
  - Perfect when needing to integrate JEE with CICS transactions
  - Subset continuing to expand

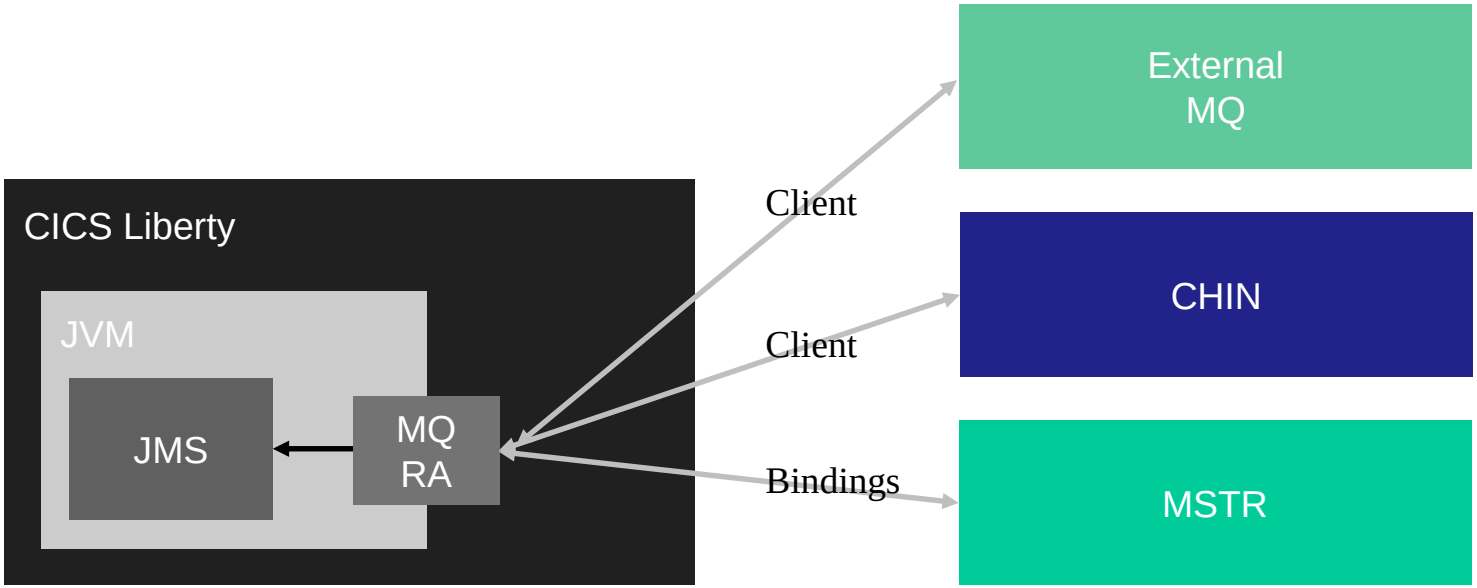
# What does this mean with MQ?

- **Allow CICS Liberty to use MQ resource adapter**
  - Just like normal Liberty
- **Run existing Liberty messaging apps (eg MDBs) inside CICS**
- **Connections to MQ supported using either client or bindings mode**
  - Some restrictions depending on environment
- **Prereqs CICS V5.3 +PI58375 +  MQ 9.0.1 RA**



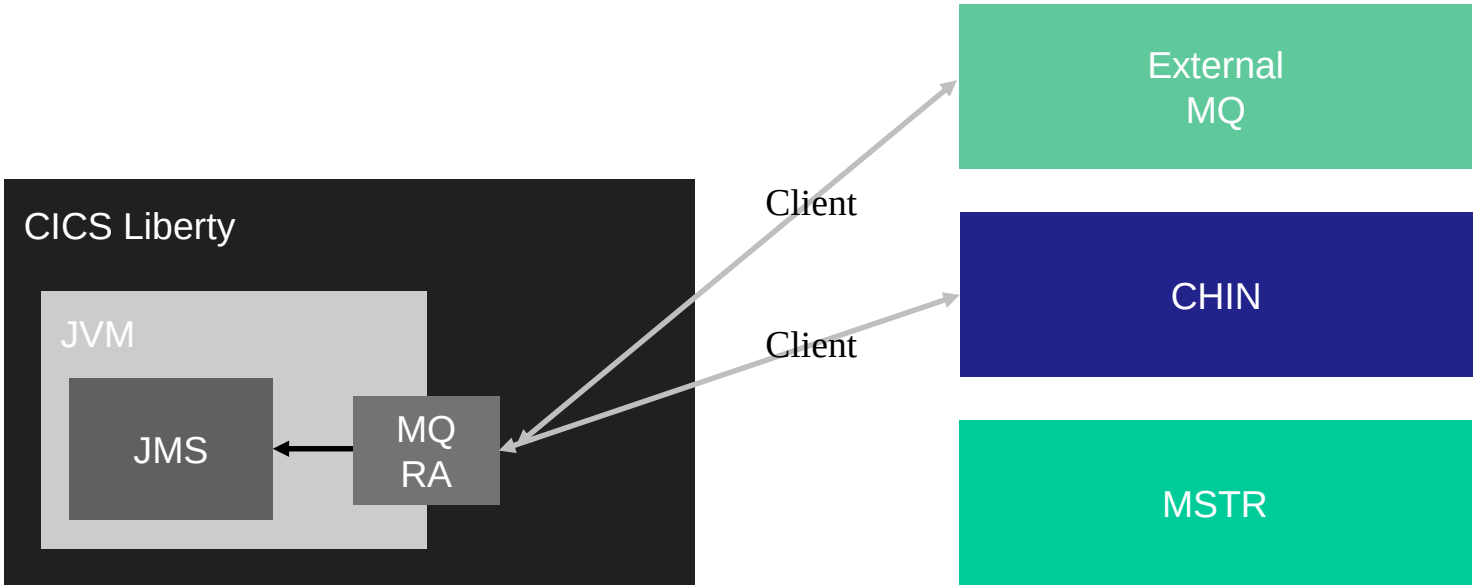
# CICS Liberty Standard JMS Support

- Bindings (RRS) and/or client (XA) connections supported
- Bindings not allowed to CICS connected MQ



# CICS Liberty Integrated JMS Support

- **Client (XA) connections only**



# Where can I find out more?

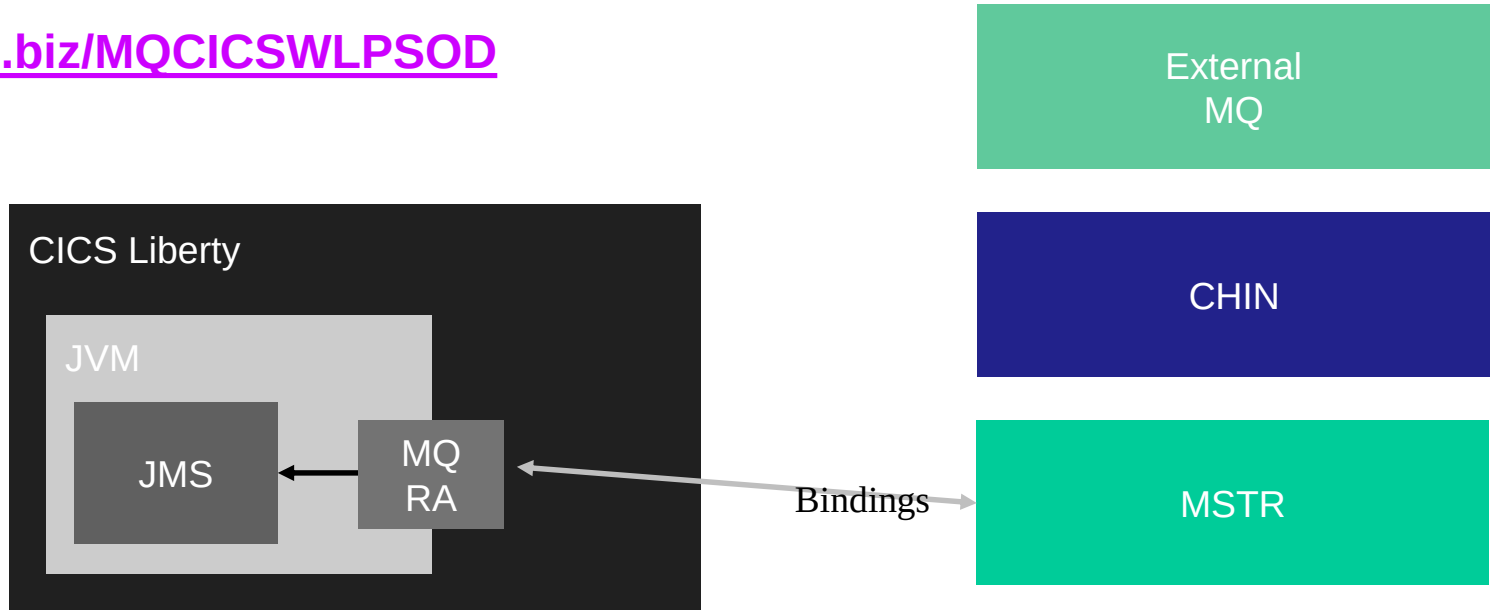
- **CICS**
  - MQ 8 KC: https://ibm.biz/MQJMSinCICSOSGiKC8
  - MQ 7.1 KC: https://ibm.biz/MQJMSinCICSOSGiKC71
  - Blog articles:
    - http://ibm.biz/MQJMSinCICSOSGiBlog
    - http://ibm.biz/MQJMSinCICSOSGiJNDI
    - https://developer.ibm.com/cics/2016/07/12/java-ee-7-in-cics/

- **IMS**
  - MQ 8 KC: https://ibm.biz/MQJMSinIMSKC
  - Blog article: https://ibm.biz/MQJMSinIMSBlog

# What is still to come?

- **Statement of direction: IBM intends to deliver additional Java EE 7 components and technologies for the CICS TS hosted WebSphere Liberty profile through continuous delivery of new features in the coming months. These additional components and technologies are intended to include support for Java EE 7 Web Profile features, support for JMS 2.0 with IBM MQ for z/OS, and the ability to LINK from a CICS TS COBOL program to a CICS TS hosted Liberty application**

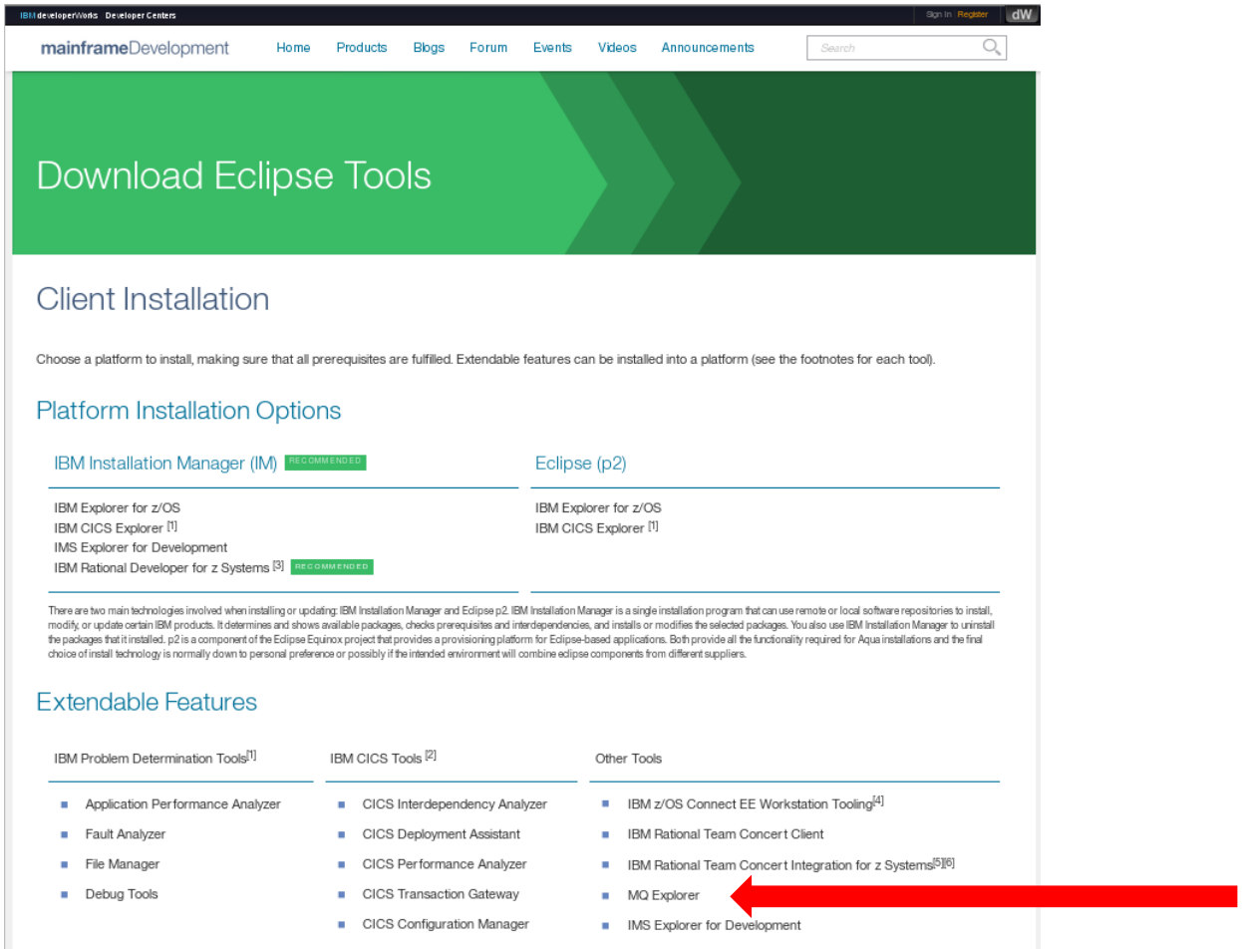
- **https://ibm.biz/MQCICSWLPSOD**



# Enhanced sub-system connectivity

# MQ Explorer is now part of Aqua

- **https://developer.ibm.com/mainframe/products/downloads/**



# CICS Adapter enhancements

- New in CICS 5.4, currently in open beta

# What we have today

- **CICS CKQC transaction controls the MQCONN definition and CKTI transactions**
  - E.g. “CKQC START” – start connection to MQ
- **Use the INITQNAME parameter on the MQCONN definition to automatically start a single instance of CKTI**
- **In order to start more than one instance of the CKTI transaction the user enters**
  - “CKQC STARTCKTI CSQ4SAMP.INITIATION.QUEUE”
  - At
    - A CICS terminal
    - An MVS console
    - Alternatively write an application program that links to program DFHMQSSQ
    - etc

# Problems with today's environment

- **If the MQCONN is restarted all transactions accessing MQ queues are terminated**
- **When the connection is re-established the transaction associated with the INITPARM parameter on the MQCONN is restarted automatically**
  - Any user initiated CKTI transactions must be manually restarted
- **The userid associated with the new CKTI transaction is that of the user issuing the restart request not the original user**
- **The default userid associated with the transaction retrieving the application message is now that of the user that issued the restart request.**
- **Cannot stop the MQ Bridge transaction without stopping / restarting the MQ connection**

# What is changing?

- **New MQMONITOR resource added**
- **Allows an associated transaction which monitors/services an MQ queue to be started/restarted automatically when the connection to the queue manager is made**
- **The transaction is automatically stopped when the queue manager is disconnected**
- **Useful for**
  - CICS trigger monitor/bridge transactions
  - Any other transaction that is dependant on MQ connectivity
- **An MQMONITOR resource is dynamically created if you specify the MQCONN INITQNAME attribute**
  - So same approach used throughout

# Resource definition

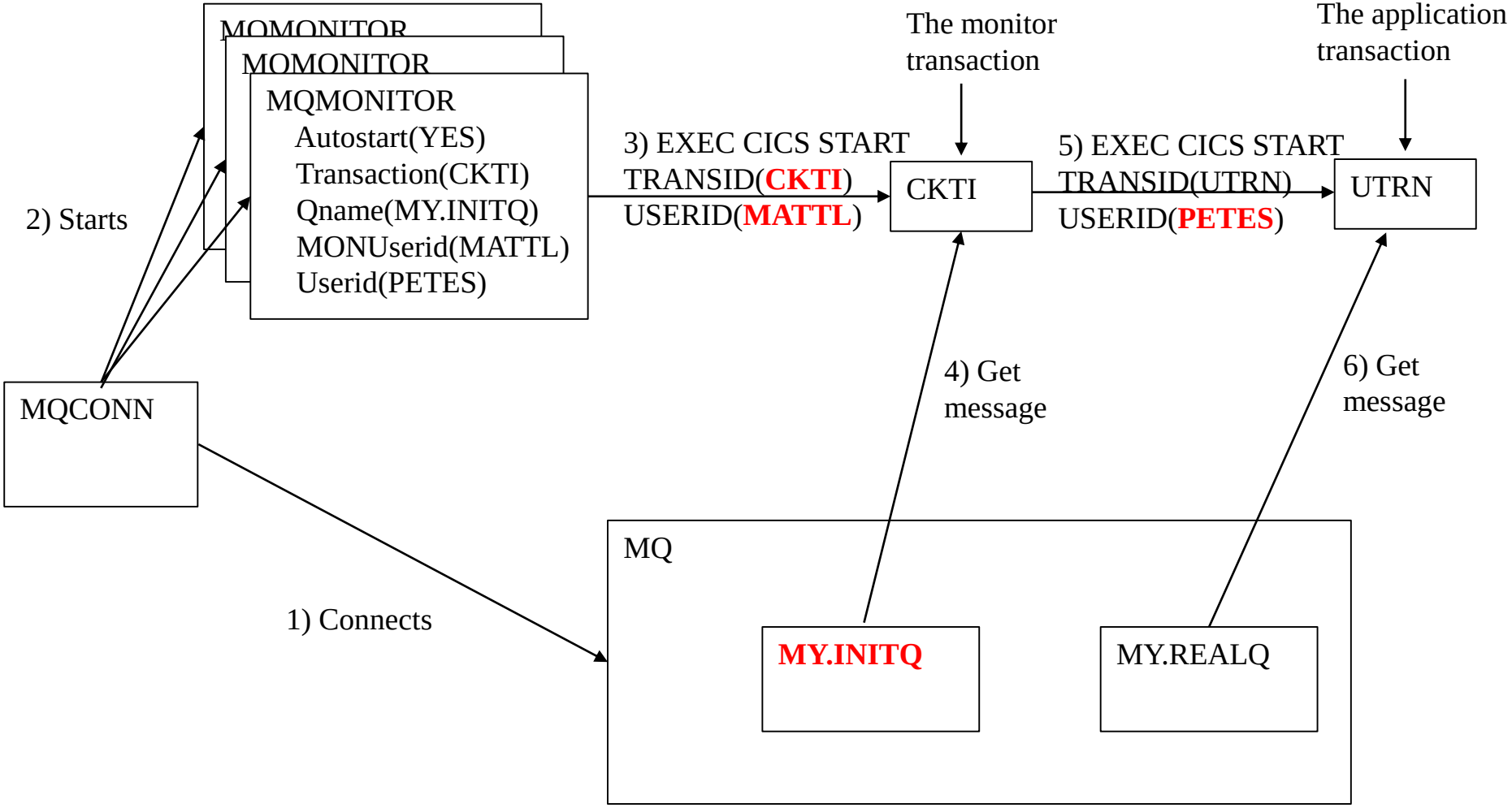
```
 OVERTYPE TO MODIFY                                          CICS RELEASE = 0710
  CEDA  ALter MQMonitor( DFHMQBR0 )
   MQMonitor       : DFHMQBR0
   Group           : MQMON
   DEScription  ==> _
   Status       ==> Enabled            Enabled | Disabled
  MONITOR ATTRIBUTES
   Autostart    ==> Yes                Yes | No
   MONData      ==> Q=FV.IYK2ZFV1.CICS.BRIDGE.QUEUE,WAIT=20,MSG=LOG
   (Mixed Case) ==>
                ==>
                ==>
                ==>
   MONUserid    ==>
   Qname        ==>
   Transaction  ==> CKBR
  APPLICATION ATTRIBUTES
   Userid       ==>
+ DEFINITION SIGNATURE

                                                      SYSID=GMB1 APPLID=IYK2ZFV1

PF 1 HELP 2 COM 3 END            6 CRSR 7 SBH 8 SFH 9 MSG 10 SB 11 SF 12 CNCL
```

- Autostart: whether started automatically when connected to MQ
- Transaction: monitoring transaction to start, defaults to CKTI
- MONUserid: userid used to start the monitoring transaction
- MONData: is arbitrary data that can be passed to the monitoring transaction
- Userid: used by the monitoring transaction for any application transactions that it starts (e.g. when CKTI starts another transaction)

# A pictorial explanation!



# CEMT SET MQCONN CONNECTED

```
 INQ MQM
 STATUS:  RESULTS - OVERTYPE TO MODIFY
  Mqm(DFHMQBR0)                    Aut                    Ena
                  Sta Tas(0000138) Tra(CKBR)
  Mqm(DFHMQINI)                    Aut                    Ena
                  Sta Tas(0000139) Tra(CKTI) Qna(IYK2ZFV1.INITIATION.QUEUE )
  Mqm(MQMQMON1)                    Aut                    Ena
                  Sta Tas(0000140) Tra(CKTI) Qna(IYK2ZFV1.INITIATION.QUEUE )
  Mqm(MQMQMON2)                    Aut                    Ena
                  Sta Tas(0000141) Tra(CKTI) Qna(CSQ4SAMP.INITIATION.QUEUE )
  Mqm(MYMQMON3)                    Aut                    Ena
                  Sta Tas(0000142) Tra(CKTI) Qna(IYK2ZFV1.PIPE4.INITQ      )
                                                    SYSID=GMB1 APPLID=IYK2ZFV1
   RESPONSE: NORMAL                                TIME: 09.05.00  DATE: 06/16/16
PF 1 HELP       3 END       5 VAR       7 SBH 8 SFH 9 MSG 10 SB 11 SF
```

# IMS MQ Connection

- **IMS V13 added Destination descriptor of type MQSERIES**
  - Use on ALT IOPCB
  - Configure in OTMA descriptor in DFSYDTx IMS PROCLIB instead of coding DFSYPRX0 or DFSYDRU0 exits
  - TYPE=MQSERIES contains parms to set MQMD values such as
    - MQRTQ  ReplyToQueue
    - MQMSGID
    - MQCORREL
    - MQPERST
    - MQFORMAT
- **APAR PI62339 (V13) PI64588 (V14) add MQSTR to allowable formats**

# DB2 MQ Connection

- **DB2 Stored Procedure Address Spaces (SPAS) connect to MQ using RRS as a transaction coordinator**
- **Each MQ operation uses a new RRS context**
  - Each RRS context has a separate thread information anchored from ACE
  - ECSA footprint of ~2kB per thread
- **MQ introduced ACELIM ZPARM to limit the number of ACEs and so ECSA consumed**
- **Initial implementation: 'protect the LPAR'**
  - Not MQ applications, which will get connection failures
- **PI48417 MQ V8, PI55066 MQ V7.1 substantially improve queue manager toleration and recovery when the limit is reached**

# Summary

- **MQ Service Provider for z/OS Connect**
  - What is z/OS Connect?
  - What is the MQ Service Provider?
  - What might it look like?

- **Other new stuff**
  - JMS enhancements
  - Enhancements to sub-system connectivity

# Questions & Answers