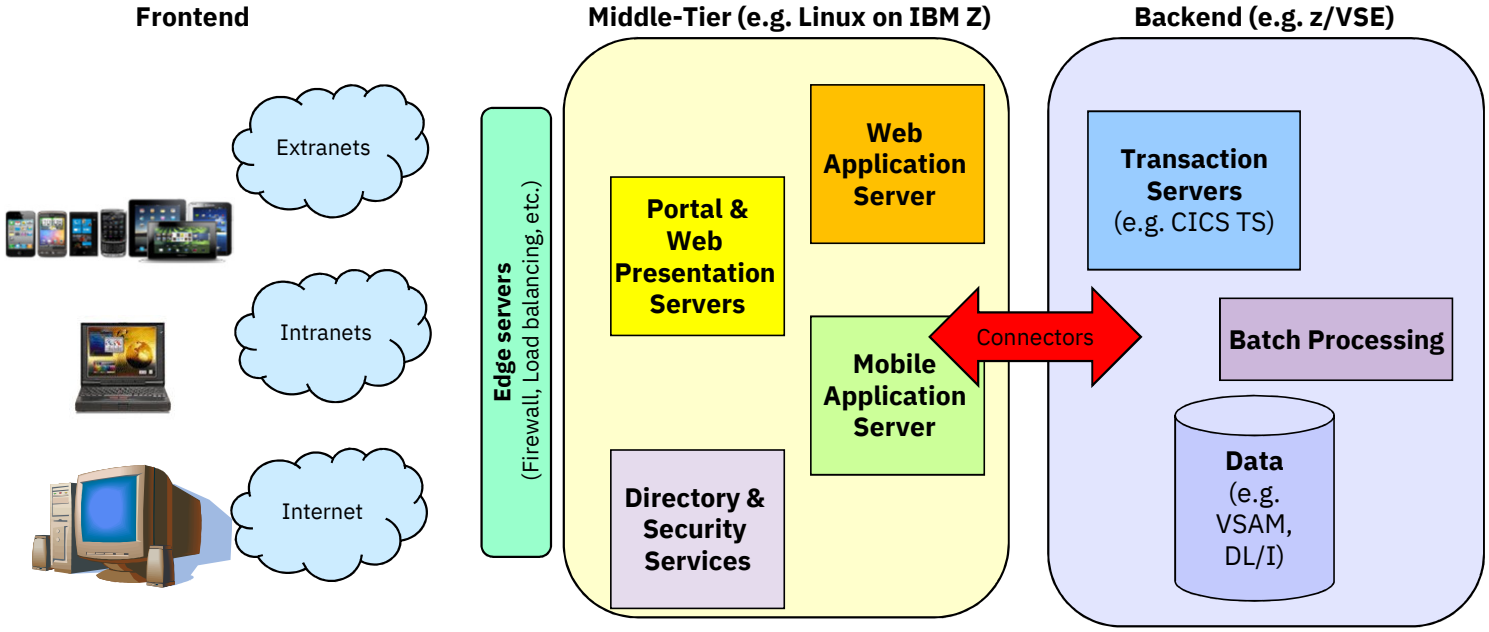
Utilizing z/VSE's REST Engine
for asynchronous messaging with IBM MQ

Ingo Franzki
ifranzki@de.ibm.com

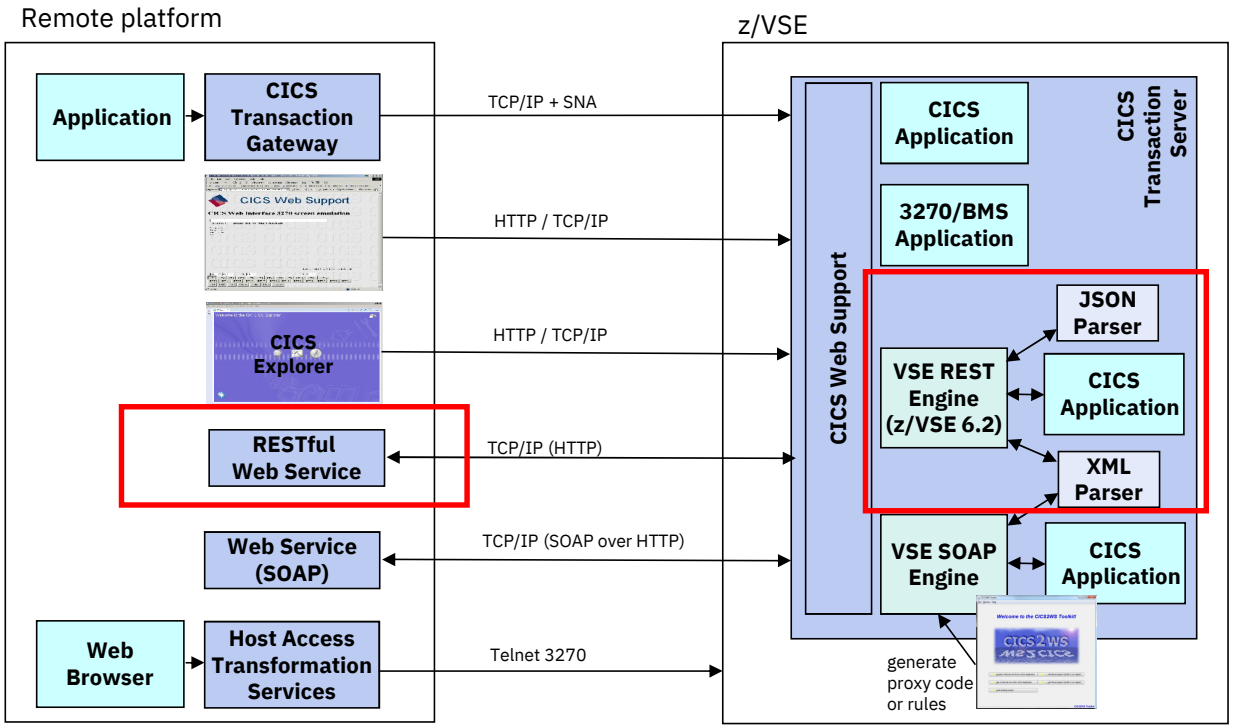
Web- / Mobile-enabling of Applications

- **Web-enable z/VSE Applications**
- **Mobile-enable z/VSE Applications**
- **Provide RESTful APIs for z/VSE Applications (microservices)**
- **Modernize User Interface for applications**



CICS Connectivity

— CICS Web Support is the base of CICS connectivity

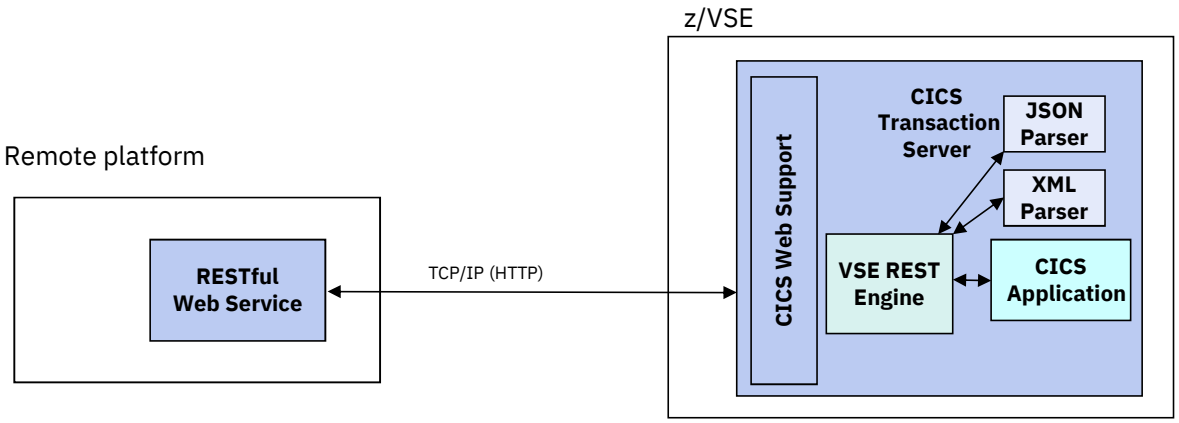


z/VSE REST Engine

z/VSE 6.2: RESTful Web Services support

— Use REST (Representational State Transfer) with CICS applications

- **Provide existing CICS applications as RESTful Web Service to the outside world**
  - z/VSE as the REST server
  - Provide an easy to use RESTful API to services for z/VSE services
- **Use/call external RESTFul Web Services from within z/VSE CICS applications**
  - z/VSE as the REST client
  - Use external RESTful APIs within z/VSE applications
- **Payload can be:**
  - JSON (JavaScript Object Notation)
  - XML
  - Plain text, Binary, Form fields, Multipart



What is REST (Representational State Transfer)?

— Representational State Transfer (REST) is a **software architecture style consisting of guidelines and best practices** for creating web services

— REST has gained widespread acceptance across the web as a **simpler alternative to SOAP** and WSDL-based web services

— RESTful systems typically communicate over the **Hypertext Transfer Protocol** (**HTTP**)
- with the same HTTP verbs (GET, POST, PUT, DELETE, and so on) used by web browsers

— The **payload** (message) transported by RESTful web services can be of various types (content types)
- Commonly used is **JSON** as well as **XML**, but it can also be plain text, or even binary data

What is REST (Representational State Transfer)?

— A RESTful web service typically operates on a certain **'object'** on a server
  - The object is typically addressed through the URI (part of the URL)
    - http://host:port/resource-uri

— Actions on such resources are typically denoted by the HTTP request types:
  - **GET** would typically **read** the resource
  - **PUT** would typically **update/replace** the resource
  - **POST** would typically **create** the resource
  - **DELETE** would typically **delete** the resource

— Additional parameters can be supplied via the URL query string
  - http://host:port/resource-uri?query-string

What is REST (Representational State Transfer)?

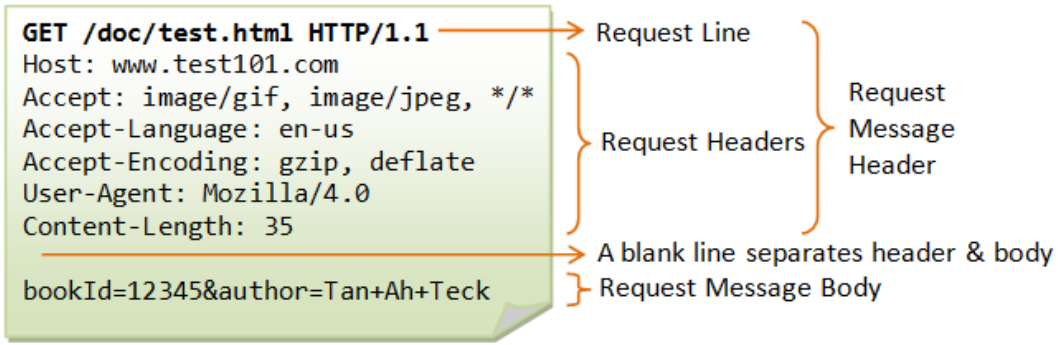
— RESTful web services are typically **stateless**
  - Each request from any client contains all the information necessary to service the request
  - The session state is therefore held in the client

— RESTful web services may use **HTTP specific features**
  - **HTTP headers** – to transport additional attributes
  - **Cookies** – to manage state information between requests

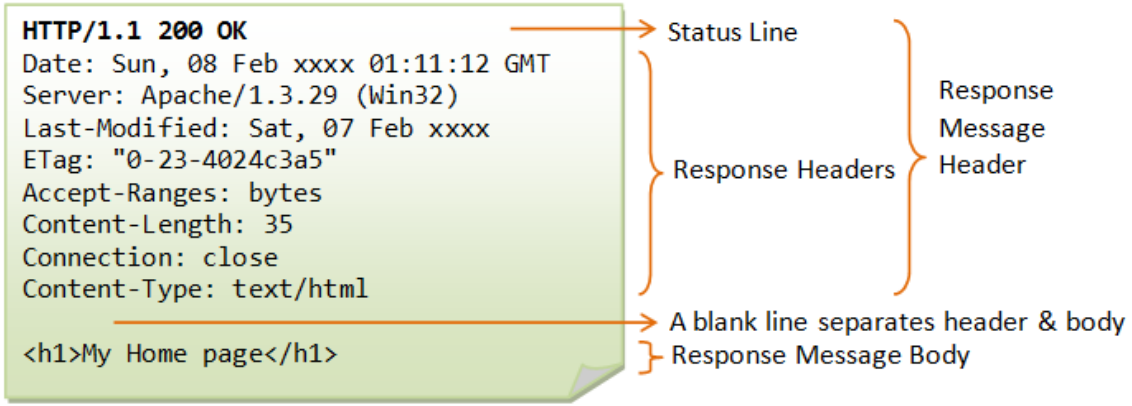
→ As denoted by the term 'typically' in above descriptions, there is no hard requirement for any of the described properties

HTTP basics

Request:



Response:



Taken from: https://www.ntu.edu.sg/home/ehchua/programming/webprogramming/HTTP_Basics.html

Example: A RESTful web service

Request a list of books:

```
Request:
GET /api/v1/books

Response:
{
  meta: {  },
  data: [{
    id: 24,
    title: 'Behavior-Driven Development',
    author: 'Viktor Farcic'
  }, {
    id: 25,
    title: 'Continuous Integration',
    author: 'Viktor Farcic'
  }]
}
```

Create a book:

```
Request:
POST /api/v1/books/id/24
{
  id: 24,
  title: 'Behavior-Driven Development',
  author: 'Viktor Farcic'
}

Response:
Status: 201 Created
{
  meta: {  },
  data: {
    uri: /api/v1/books/id/24
  }
}
```

Request a single book:

```
Request:
GET /api/v1/books/id/24

Response:
{
  meta: {  },
  data: {
    id: 24,
    title: 'Behavior-Driven Development',
    author: 'Viktor Farcic'
  }
}
```

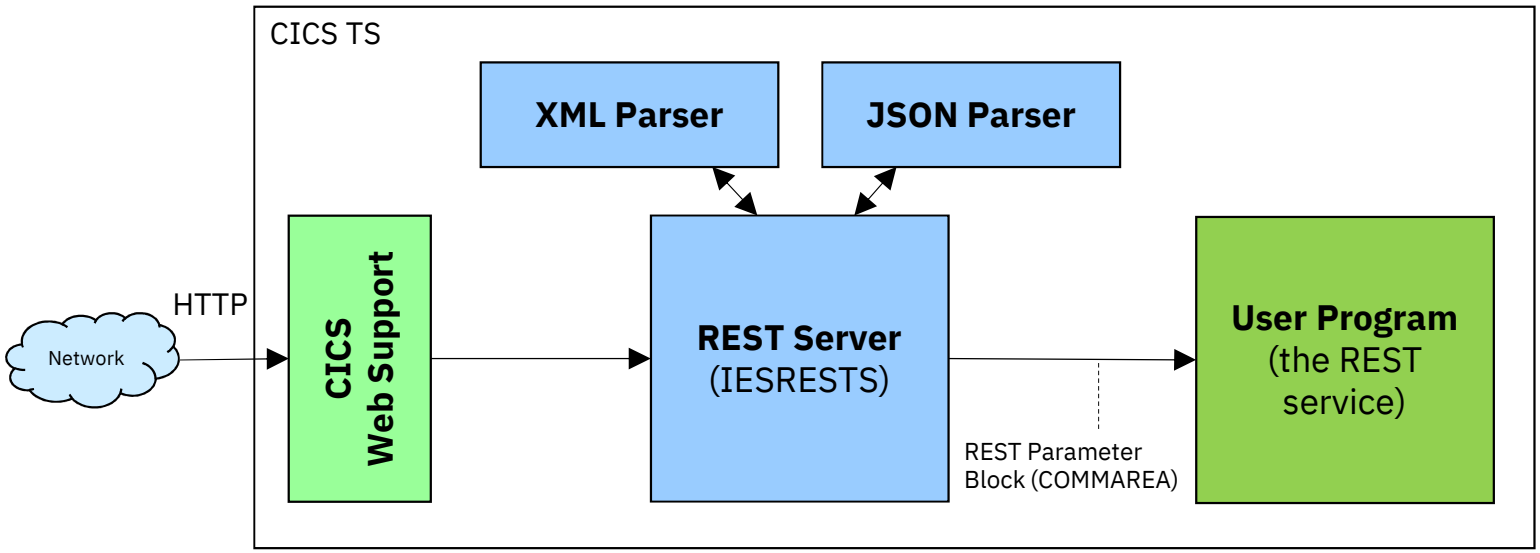
Delete a book:

```
Request:
DELETE /api/v1/books
{
  id: 24,
}

Response:
Status: 202 Accepted
{
  meta: {  },
  data: {  }
}
```

Example taken from: https://technologyconversations.com/2014/08/12/rest-api-with-json/

z/VSE 6.2: z/VSE as a REST Server

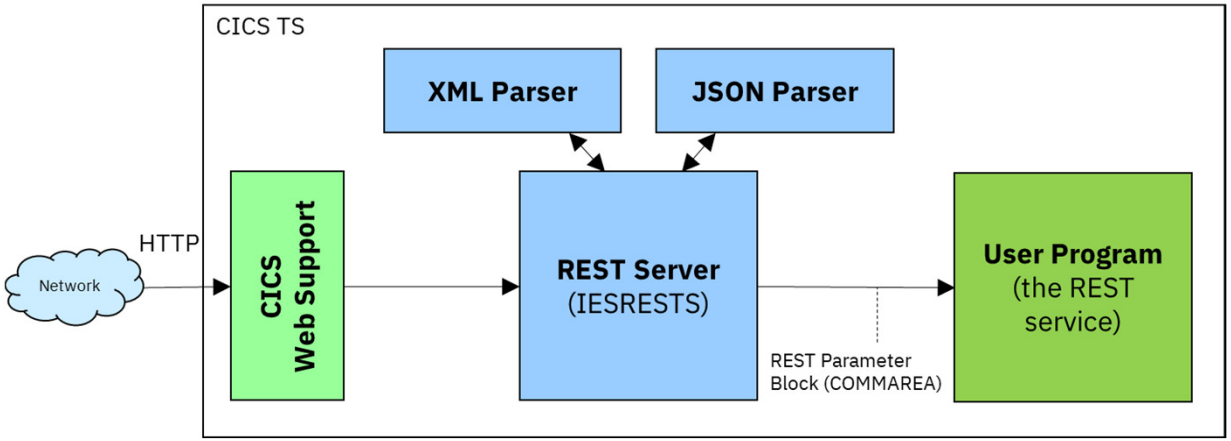


Description of the REST Parameter Block:
https://www.ibm.com/support/knowledgecenter/SSB27H_6.2.0/fa2ws_how_rest_control_blocks_are_used.html

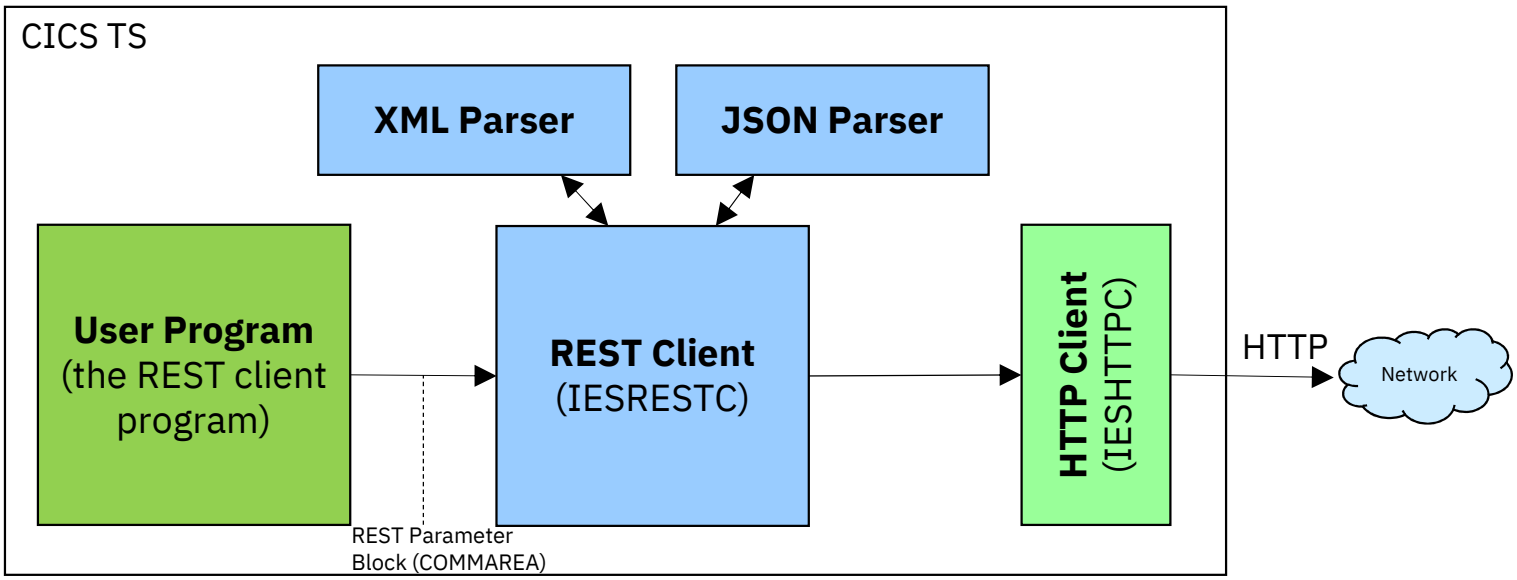
z/VSE 6.2: z/VSE as a REST Server

— The z/VSE REST-Engine...

- Receives the request (via CICS Web Support)
- Extracts information from the request:
  - User program to call from the URL:
    http://host:port/cics/CWBA/IESRESTS/user-program/resource-uri[?query-string]
  - URL parameters from the query string (if any)
  - HTTP headers
  - Cookies (if any)
  - Request data (if any)
- Calls the user program
- Constructs the response:
  - HTTP status code
  - HTTP headers
  - Set-Cookies requests (if any)
  - Response data (if any)
- Sends the response back to the client



z/VSE 6.2: z/VSE as a REST Client

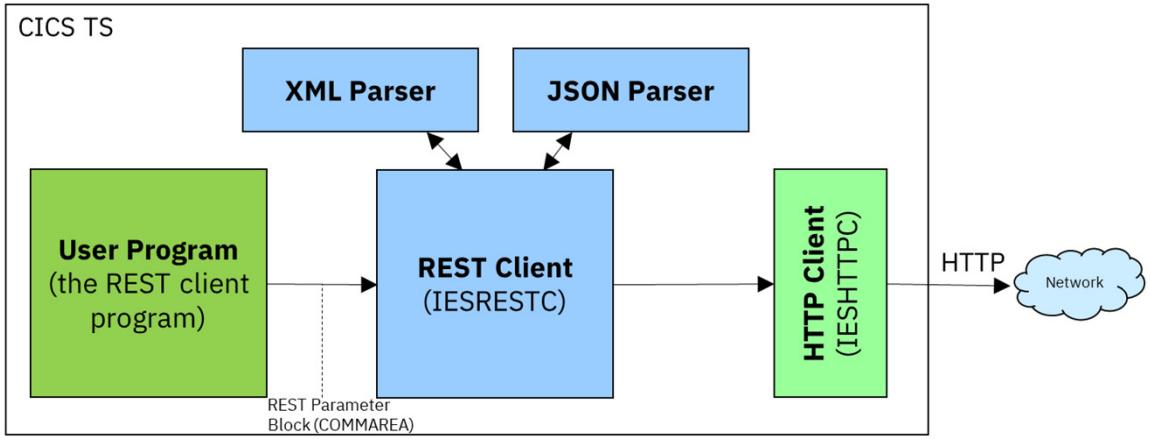


Description of the REST Parameter Block:
https://www.ibm.com/support/knowledgecenter/SSB27H_6.2.0/fa2ws_how_rest_control_blocks_are_used.html

z/VSE 6.2: z/VSE as a REST Client

— The z/VSE REST-Engine...

- Gets called from the user program
- Constructs the request
  - Splits the URL into host and port and resource-uri
    http://host:port/resource-uri[?query-string]
  - Adds URL parameters to the query string (if any)
  - HTTP headers
  - Cookies (if any)
  - Request data (if any)
- Sends the request to the server
- Receives the response from the server
- Extracts information from the response:
  - HTTP status code
  - HTTP headers
  - Set-Cookies requests (if any)
  - Response data (if any)
- Returns back to the user program



The REST parameter block

Contains information about the request

- Request type (GET, PUT, POST, ...)
- URL
- Content-Type
- Data-type (XML, JSON, plain text, binary)
- URL parameters from query string (http://.....?a=b&c=d)
- Form fields
- HTTP headers
- Cookies
- Authentication information
- Response status code

Copybooks in PRD1.BASE:

- IESRESTH.H LE/C
- IESRESTL.C COBOL
- IESJSONP.P PL/1
- IESRESTA.A HLASM

```
* COMMAREA layout used by the REST Engine to call the user program
* (VSE as REST server) or to get called by the user program
* (VSE as REST client):
*
 01 REST-COMMAREA.
   02 REST-VERSION               PIC 9(9) BINARY.
   02 REST-EBCDIC-CODEPAGE       PIC X(16).
   02 REST-FLAGS                 PIC 9(9) BINARY.
   02 REST-RETCODE               PIC 9(9) BINARY.
   02 REST-PRIVATE               USAGE IS POINTER.
*  Request specific fields:
   02 REST-REQ-ACTION            PIC 9(9) BINARY.
   02 REST-REQ-URL               PIC X(2048).
   02 REST-REQ-CONTENT-TYPE      PIC X(128).
   02 REST-REQ-DATA-TYPE         PIC 9(9) BINARY.
   02 REST-REQ-DATA-PTR          USAGE IS POINTER.
   02 REST-REQ-DATA-LENGTH       PIC 9(9) BINARY.
   02 REST-REQ-URL-PARAMS-TSQ    PIC X(8).
   02 REST-REQ-FORM-FIELDS-TSQ   PIC X(8).
   02 REST-REQ-HTTP-HEADERS-TSQ  PIC X(8).
   02 REST-REQ-COOKIES-TSQ       PIC X(8).
   02 REST-REQ-AUTH-TYPE         PIC 9(9) BINARY.
   02 REST-REQ-AUTH-USER         PIC X(64).
   02 REST-REQ-AUTH-PASSWORD     PIC X(64).
   02 REST-REQ-ACCEPT            PIC X(128).
*  Response specific fields:
   02 REST-RESP-HTTP-STATUS-CODE PIC 9(9) BINARY.
   02 REST-RESP-HTTP-STATUS-TEXT PIC X(128).
   02 REST-RESP-CONTENT-TYPE     PIC X(128).
   02 REST-RESP-DATA-TYPE        PIC 9(9) BINARY.
   02 REST-RESP-DATA-PTR         USAGE IS POINTER.
   02 REST-RESP-DATA-LENGTH      PIC 9(9) BINARY.
   02 REST-RESP-FORM-FIELDS-TSQ  PIC X(8).
   02 REST-RESP-HTTP-HEADERS-TSQ PIC X(8).
   02 REST-RESP-COOKIES-TSQ      PIC X(8).
   02 REST-RESP-LOCATION         PIC X(2048).
```

Description of the REST Parameter Block:
https://www.ibm.com/support/knowledgecenter/SSB27H_6.2.0/fa2ws_how_rest_control_blocks_are_used.html

Handling XML and JSON data

The z/VSE REST Engine automatically translates request and response data

- **XML:** Content-Type: text/xml or application/xml
  - XML data is parsed by the XML parser
  - An XML tree in memory is passed to the user program
- **JSON:** Content-Type: text/json or application/json
  - JSON data is parsed by the JSON parser
  - A JSON tree in memory is passed to the user program
- **URL encoded:** Content-Type: application/x-www-form-urlencoded
  - Form field data is parsed and passed via TS queue entries
- **Plain text:** Content-Type: text/* (other than xml or json)
  - ASCII-EBCDIC converted
- **Binary data:** anything else
- **Multipart data:**
  - Each part is converted individually based on its content type

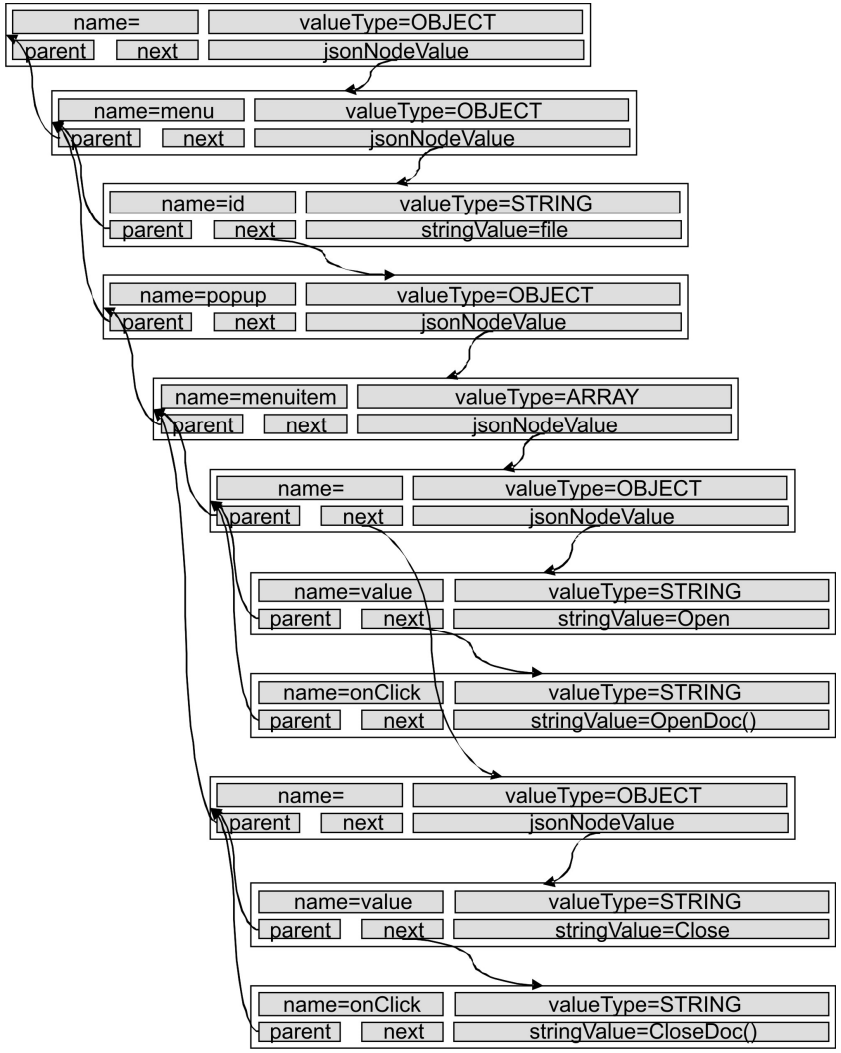
JSON data

Example:

```
{"menu": {
  "id": "file",
  "popup": {
    "menuitem": [
      {"value": "Open", "onclick": "OpenDoc()"},
      {"value": "Close", "onclick": "CloseDoc()"}
    ]
  }
}}
```

The JSON control blocks are defined in copybooks in PRD1.BASE:

- IESJSONH.H LE/C
- IESJSONC.C COBOL
- IESJSONP.P PL/1
- IESJSONA.A HLASM



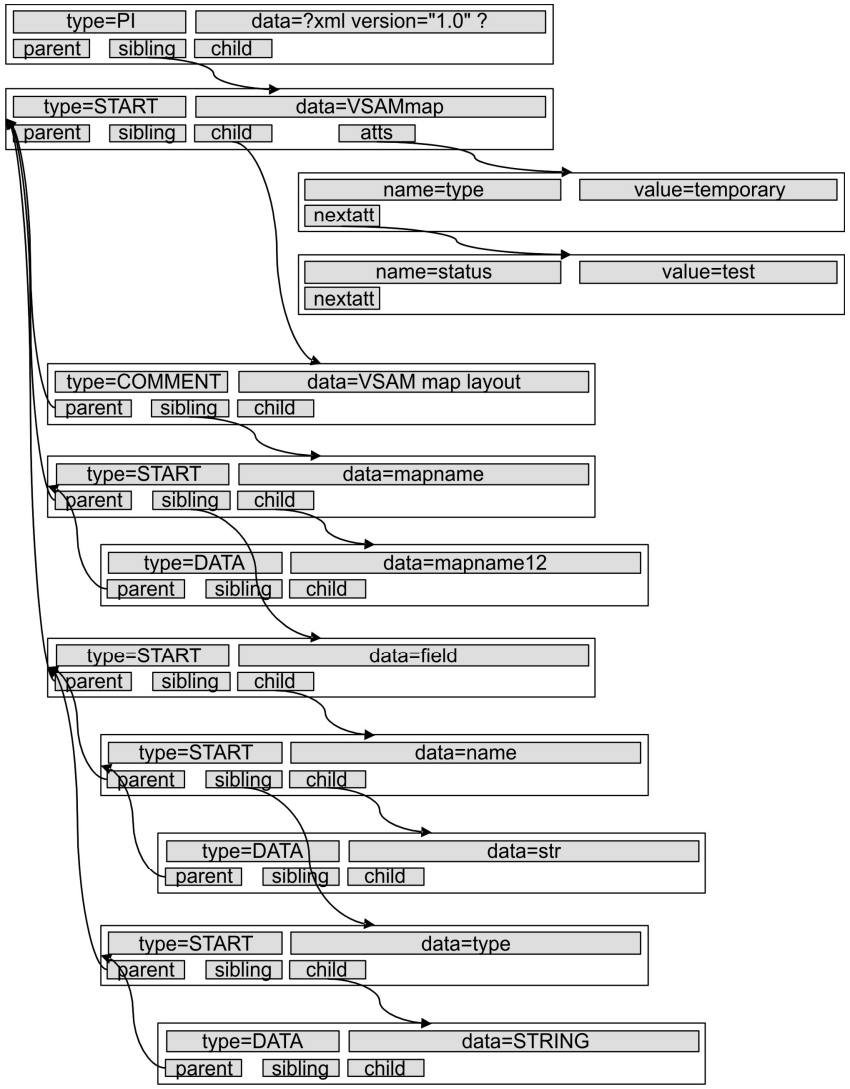
XML data

Example:

```
<?xml version="1.0" encoding="UTF-8"?>
<VSAMmap type="temporary" status="test">
 <!-- VSAM map layout -->
 <mapname>mapname12</mapname>
 <field>
  <name>str</name>
  <type>STRING</type>
 </field>
 <field>
  <name>sign</name>
  <type>SIGNED</type>
 </field>
</VSAMmap>
```

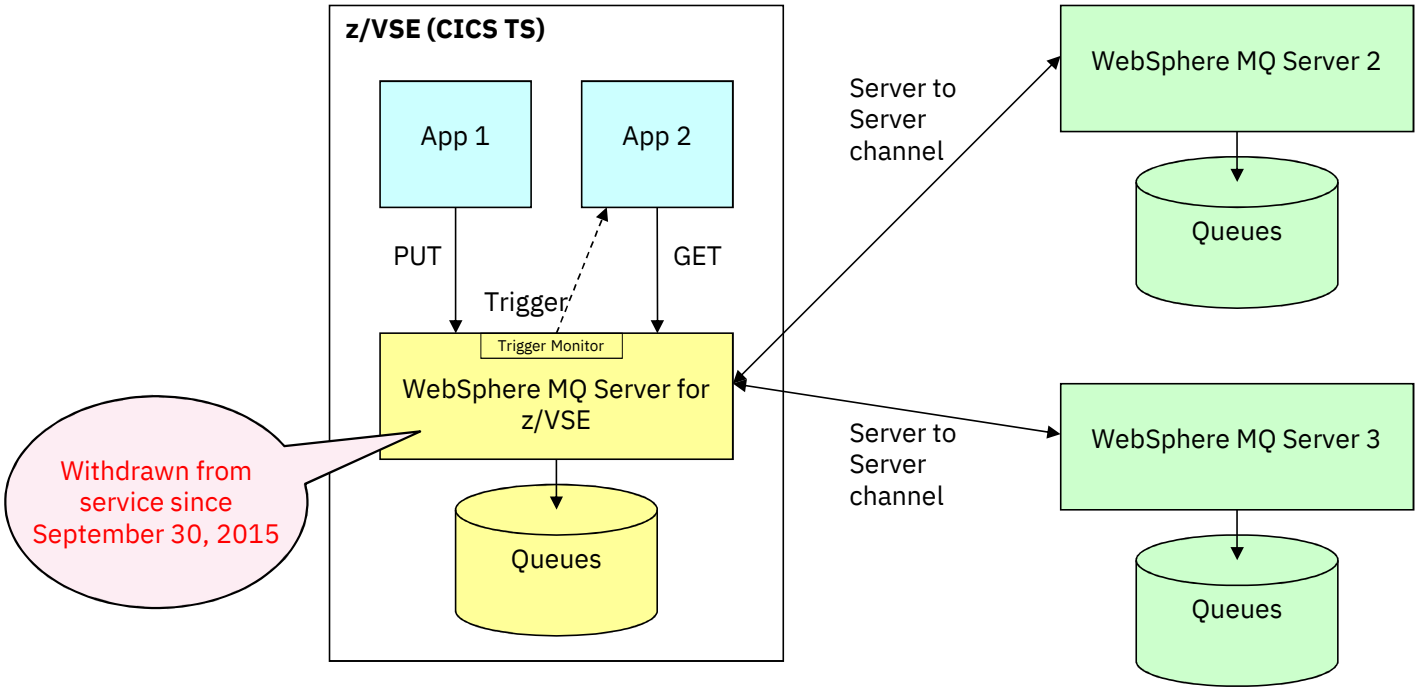
The XML control blocks are defined in copybooks in PRD1.BASE:

- IESXMLAH.H LE/C
- IESXMLCB.C COBOL
- IESXMLPL.P PL/1
- IESXMLAS.A HLASM



Asynchronous messaging with IBM MQ

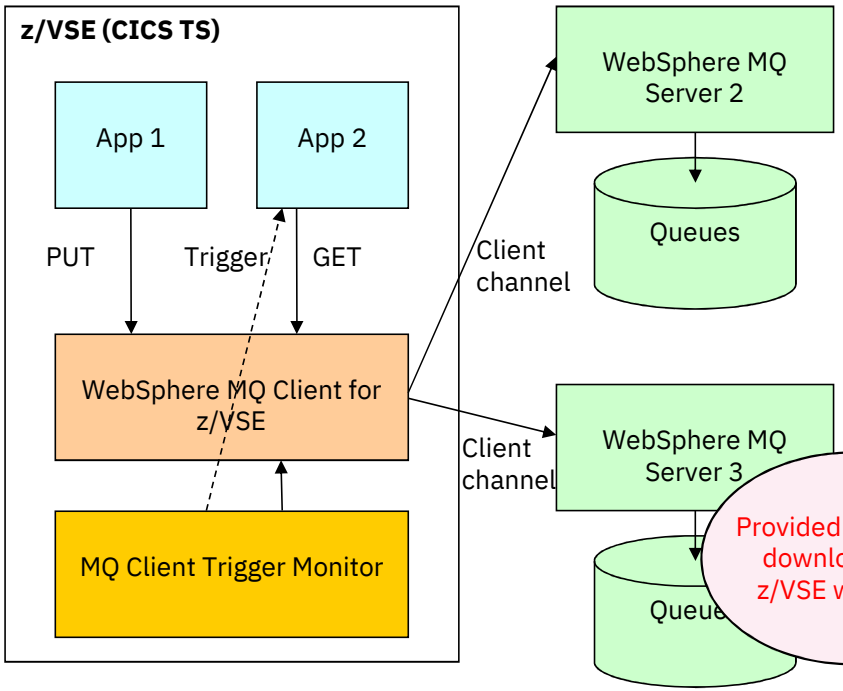
WebSphere MQ Server for z/VSE V3.0



Note:
- z/VSE Applications don‘t have to be aware of MQ Server 2 and 3
- Decision which MQ Server Queue will host the messages is defined in channels between the MQ servers

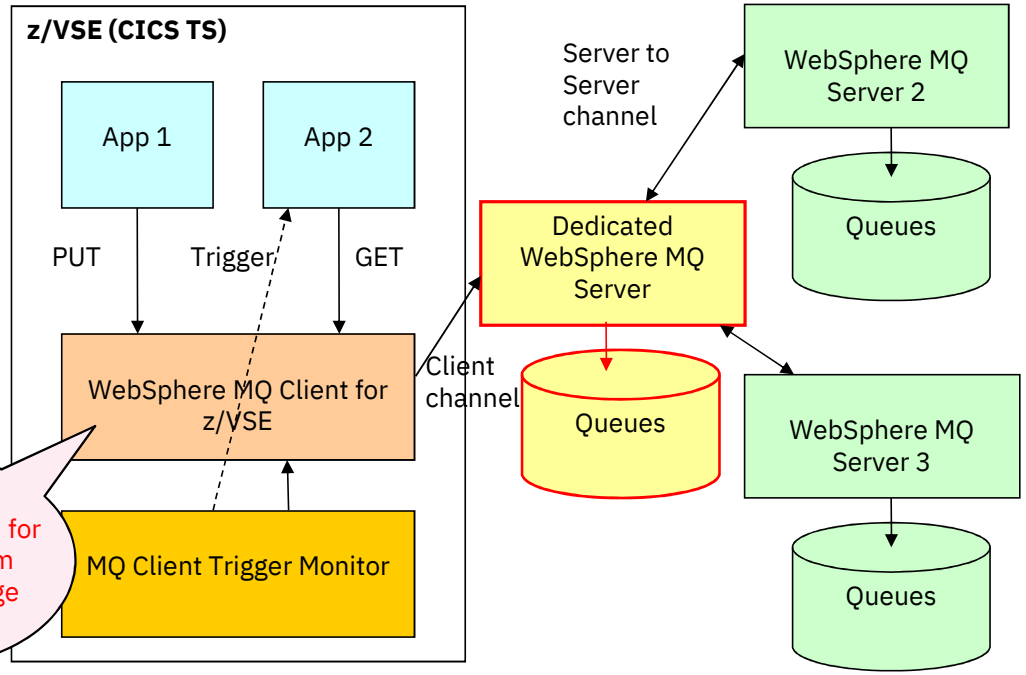
Replace the MQ Server on z/VSE with the MQ Client on VSE

Option 1:
z/VSE applications work directly with the MQ Servers 2 and 3



Note:
- z/VSE Applications have to be aware of MQ Server 2 and 3
- Decision which MQ Server Queue will host the messages has to be in the application

Option 2:
Add a dedicated MQ Server (i.e. on Linux on System z)



Note:
- z/VSE Applications are all working with the same MQ Server
- All definitions from z/VSE MQ Server can be replicated to the new dedicated MQ server
- z/VSE applications need to be aware of just one MQ server

IBM MQ V9.1

— **From the Announcement letter ZP18-0303 from July 3, 2018**

- New capabilities for both MQ base and MQ Advanced clients
  - ...
  - **REST API for messaging:**
    As more programming becomes reliant on using RESTful APIs, some developers need to use the same approach for sending and receiving messaging data.
    In MQ V9.1, there is support for using the MQ REST API for point-to-point messaging.

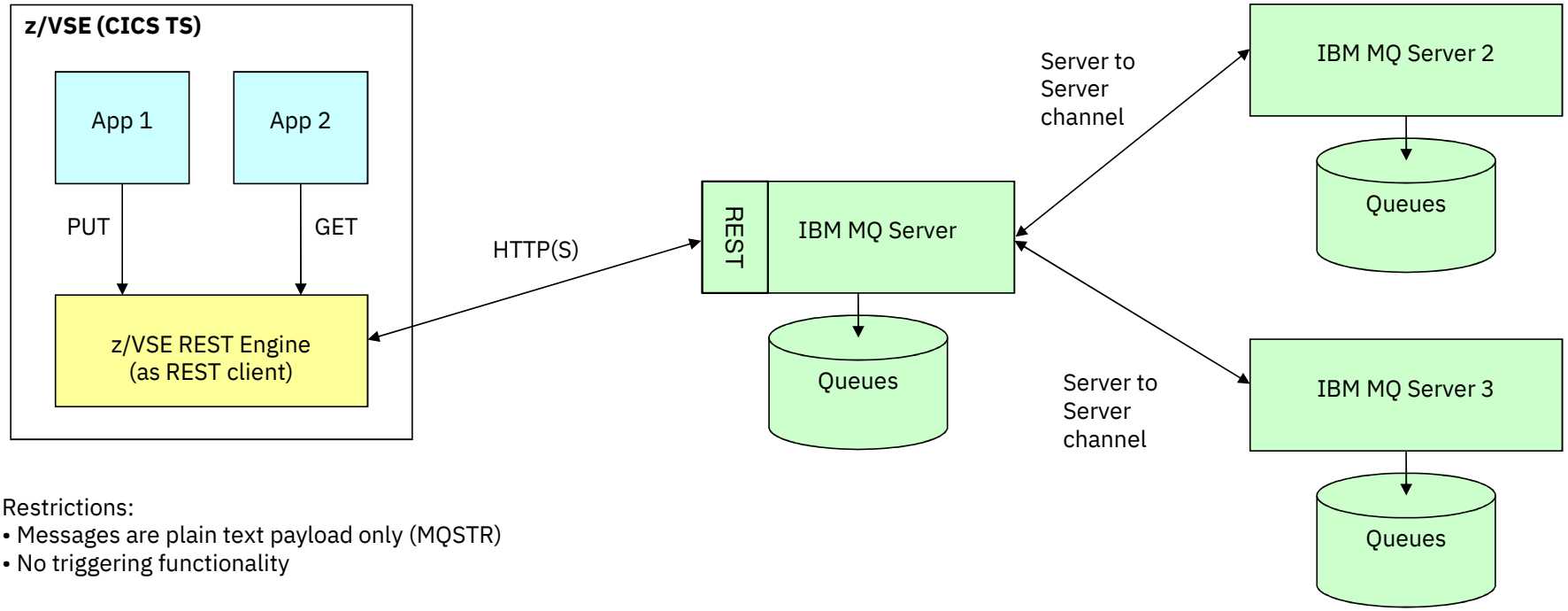
— **A simple light-weight, built-in, REST API for messaging**

- Doesn't require installation of an MQ client
- Allows you to build messaging into your applications regardless of where they run, or the language they are written in

— **Currently supports point-to-point text based messaging**

— **Evolution of the messaging REST API will continue in 9.1.x CD**

IBM MQ with REST API for Messaging



Restrictions:
- Messages are plain text payload only (MQSTR)
- No triggering functionality

→ Fully supported environment with z/VSE 6.2

IBM MQ REST API for Messaging - Put a message onto a queue

— **HTTP POST /ibmmq/rest/v1/messaging/qmgr/{qmgrName}/queue/{queueName}/message**

— **Request body must be text and use UTF-8 encoding**
- No specific text structure is required
- An MQSTR formatted message containing the request body text is created and put to the specified queue

— **Additional HTTP headers (optional)**

• Authorization	e.g. for HTTP basic authentication
• ibm-mq-md-correlationId	Correlation ID of the created message
• ibm-mq-md-expiry	Expiry duration for the created message (default: unlimited)
• ibm-mq-md-persistence	Persistence for the created message (default: nonpersistent)
• ibm-mq-md-replyTo	Reply-to destination for the created message

— **Response code indicates success or failure**

• 201	Message created and sent successfully
• 4xx/5xx	Error creating the message

— **Response HTTP headers:**

• ibm-mq-md-messageId	Message ID that is allocated by IBM MQ to this message

IBM MQ REST API for Messaging - Get a message from a queue

- **HTTP DELETE /ibmmq/rest/v1/messaging/qmgr/{qmgrName}/queue/{queueName}/message**
- **Query parameters (optional)**
  - correlationId=*hexValue* Return the next message with the corresponding correlation ID
  - messageId=*hexValue* Return the next message with the corresponding message ID
  - wait=*integerValue* Wait *integerValue* milliseconds for the next message to become available
- **Additional HTTP headers (optional)**
  - Authorization e.g. for HTTP basic authentication
- **Response body is the message content as text and use UTF-8 encoding**
- **Response code indicates success or failure**
  - 200 Message received successfully
  - 204 No message available
  - 4xx/5xx Error receiving the message
- **Response HTTP headers:**
  - ibm-mq-md-correlationId Correlation ID of the received message
  - ibm-mq-md-expiry Expiry duration for the received message
  - ibm-mq-md-persistence Persistence for the received message
  - ibm-mq-md-replyTo Reply-to destination for the received message

Authentication options with IBM MQ's REST API for Messaging

— **No authentication**
- Not recommended

— **HTTP basic authentication**
- User-id and password is sent as part of the HTTP request
- Should use HTTP over SSL/TLS (HTTPS)

— **Token based authentication**
- Requires a **Login** request (with user-id and password) before any other messaging API calls can be made
- MQ returns an LTPA token that needs to be passed with any further requests
- Optional **Logout** request
- Should use HTTP over SSL/TLS (HTTPS)

— **SSL/TLS client authentication**
- SSL/TLS client certificate is used for authenticating the user
- Requires the use of HTTPS

Whitepaper and example programs

— **Whitepaper:**
- ftp://public.dhe.ibm.com/eserver/zseries/zos/vse/pdf3/VSE2RESTMQ.pdf

— **Example programs (COBOL):**
- ftp://public.dhe.ibm.com/eserver/zseries/zos/vse/download/xmps/MQ_via_REST.zip

— **IBM MQ Messaging REST API description:**
- https://www.ibm.com/support/knowledgecenter/SSFKSJ_9.1.0/com.ibm.mq.ref.adm.doc/q127980_.htm

Questions?

Notices and disclaimers

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