Migrating from MQ Server on z/VSE to MQ Client using the z/VSE MQ Client Trigger Monitor

Last formatted on: Tuesday, February 07, 2017

Ingo Franzki
ifranzki@de.ibm.com

Migrating from MQ Server on z/VSE to MQ Client

This document explains how to migrate an application environment that is currently using WebSphere MQ Server for z/VSE V3.0.0 (5655-U97) to WebSphere MQ Client for VSE.

The IBM product "WebSphere MQ for z/VSE V3.0" (5655-U97) is withdrawn from marketing on September 8, 2014. and it will be withdrawn from service effective September 30, 2015. WebSphere MQ for z/VSE V3.0 implements an MQ Server that supports a wide range of typical MQ functions, such as queuing, triggering, publish and subscribe channels, etc.

The EoM announcement letter 914-104 contains the following note:

> *While IBM WebSphere MQ for z/VSE V3.0 is being withdrawn from marketing with no direct replacement, a WebSphere MQ Client for VSE will continue to be available to enable connectivity over WebSphere MQ from VSE environments to MQ deployments on other systems.*

Thus, one alternative option for users of WebSphere MQ for z/VSE V3.0 is to use the MQ Client on VSE instead, which is available as SupportPac MQC5 for download from http://www.ibm.com/support/docview.wss?rs=171&uid=swg24010051&loc=en_US&cs=utf-8&lang=en

From the README.TXT file contained in the MQC5 package:

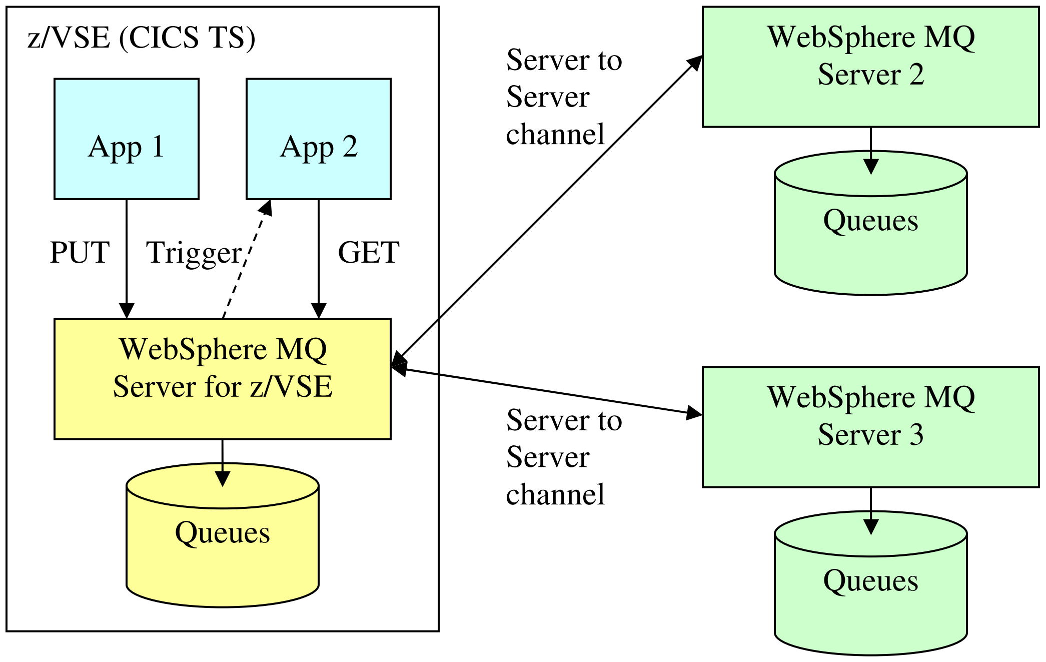
> *A WebSphere MQ client is part of the product that can be installed on its own, on a separate machine from the base product and server. You can run a WebSphere MQ (MQSeries) application on a WebSphere MQ client and it can interact, by means of a communications protocol, with one or more WebSphere MQ servers and connect to their queue managers. The servers to which the client connects may or may not be part of a cluster. WebSphere MQ Family Platforms provides a list of which platforms are supported as WebSphere MQ Clients and/or Servers.*

This document contains 3 main chapters:

1. An overview chapter that describes a typical application environment WebSphere MQ Server for z/VSE V3.0.0 (5655-U97) and how the environment would look like when using MQ Client instead..
2. A chapter that describes the setup of WebSphere MQ Client for z/VSE and what changes are required in the applications.
3. A chapter that describes the z/VSE function MQ Client Trigger Monitor that allows using triggering with MQ Client applications.

Chapter 1: Overview

This chapter describes a typical application environment WebSphere MQ Server for z/VSE V3.0.0 (5655-U97).



The WebSphere MQ Server for z/VSE runs inside a CICS region. It allows CICS or batch applications to use the MQ programming interface (API) to connect to the queue manager, open message queues, and put messages onto queues, or get messages from queues.

Applications can be triggered on arrival from messages on certain queues. This means, that when messages arrive on a queue (which has triggering configured), that the MQ Server starts/invokes/calls a so called trigger-program. The trigger program (App2 in above example) then opens the queue that it was triggered for and gets messages from the queue and processes them.

One MQ server may be interconnected to other MQ servers through server-to-server channels over a TCP/IP network. This allows that one MQ server passes messages from a (transmission-) queue to a queue on another MQ server. The server-to-server channels are typically activated when one or a certain amount of messages appear on the transmission queue. Then all messages are transmitted (in batches), and only after they have been successfully stored on the receiving queue, they are deleted from the transmission queue.

Example application scenario

To get a better understanding of the whole environment, let's take a closer look at an example application scenario. This scenario is based on the picture above.

- App 1 puts a message onto a local queue of the MQ Server on VSE.
- This local queue is configured as a transmission queue, and thus, the server-to-server channel to MQ Server 2 (e.g. running on a distributed system) is started and the message is transferred to a queue on MQ Server 2.
- A trigger is defined on that queue, which starts an application on MQ Server 2.
- This trigger application gets the message from the queue and processes it.
- A response message is put onto another queue on MQ Server 2.
- That queue is also configured as a transmission queue, and thus, the server-to-server channel to the MQ Server on z/VSE. Thus, the response message is transferred back to a queue on MQ Server on VSE.
- A trigger is defined on that queue, which starts a application App2 on z/VSE.
- App 2 gets the reply message from the queue, and processes it.

Of course, the scenario could be much more complex, e.g. involve MQ Server 3 as well, or even transform the message and pass it to other systems.

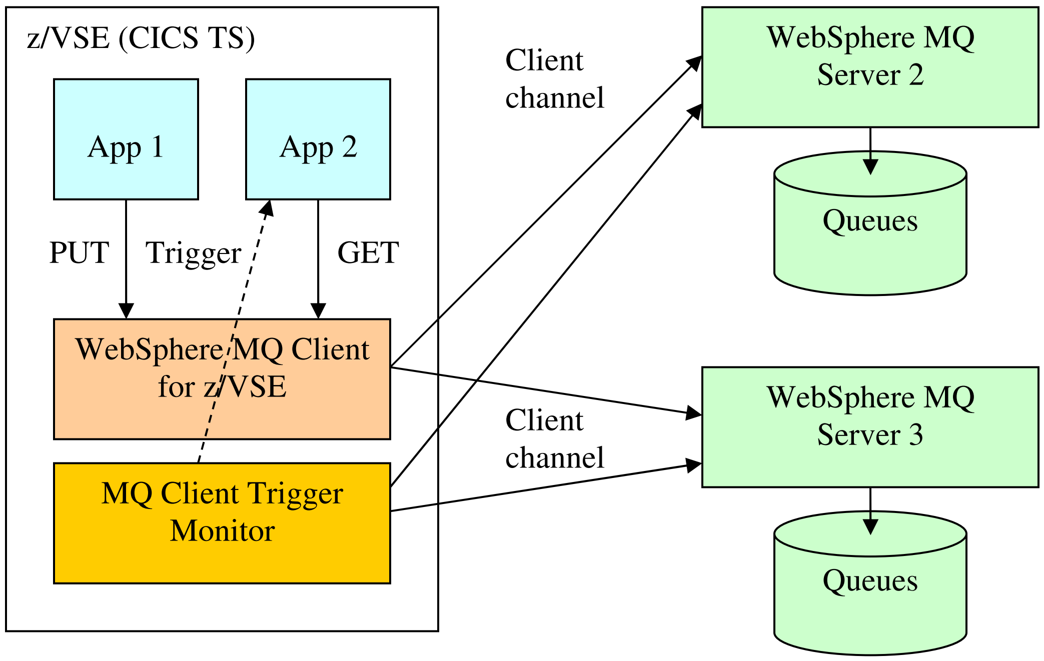
Using MQ Client

When using the MQ client on z/VSE instead of the MQ Server, a few environmental changes must be made.

With MQ Client, there are no queues locally on z/VSE any more. Instead the applications work remotely (through the MQ Client) with queues that are located on another MQ Server on another system.

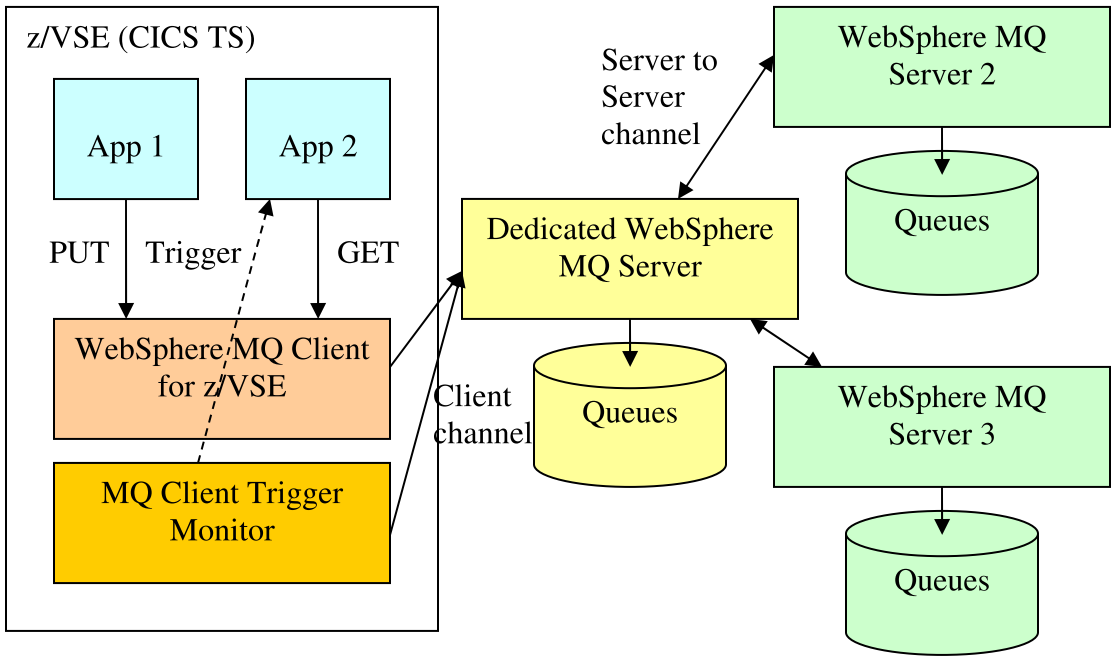
There are 2 options to migrate the environment shown in the picture above to an environment using MQ Client:

1. **Replace the MQ Server on z/VSE with the MQ Client on VSE and let the VSE applications work directly with the MQ Servers 2 and 3.**



Or:

2. **Add a dedicated MQ Server with which the applications on z/VSE work through the MQ Client. This dedicated MQ Server then has server-to-server channels to the other MQ Servers in the environment.**



While option 1 results in a simpler environment, it may require more changes in the z/VSE applications that access MQ queues, especially when multiple MQ Servers are used. Here, the z/VSE applications may need to know/select the MQ Server they want to work with. The applications may also need to use different queue names, to access the queues of MQ Servers 2 and 3. In addition, configuration changes are required on MQ Server 2 and 3 to allow client applications to connect, and to remove the server-to-server channel to z/VSE.

Option 2 adds a dedicated MQ Server (may be shared between multiple z/VSE systems), so that the z/VSE applications always work with the same MQ Server. Here the configuration of MQ Server 2 and 3 can stay pretty much the same (may need different IP addresses or hostnames for the channels). The dedicated server may have the same queues that the MQ Server on z/VSE used to have. Thus changes to the VSE applications are minimal.

Which of the options you choose (or a mixture of both), depends on the complexity of the environment, the applications, and the use cases.

Chapter 2: Application changes to MQ Client for z/VSE

This chapter describes what steps are required when migrating the existing applications to use MQ Client instead of MQ Server.

Installing the WebSphere MQ Client for z/VSE

The WebSphere MQ Client SupportPac can be downloaded from http://www.ibm.com/support/docview.wss?rs=171&uid=swg24010051&loc=en_US&cs=utf-8&lang=en

Category 3 WebSphere MQ SupportPacs are supplied under the standard terms and conditions provided by the International Program License Agreement (IPLA) and thus, carry program defect service for WebSphere MQ customers. Please read the IPLA and LI files that accompany the SupportPac, to ensure you understand the conditions under which the SupportPac is provided.

If you encounter what you believe to be a defect with the SupportPac, and you have a current license for a WebSphere MQ server product, you may request Program Services by reporting the problem via the same defect reporting channel you employ for the WebSphere MQ server product(s) on which you are using the SupportPac. No proof of entitlement is required to use this SupportPac.

Following the End of Support announcement for WebSphere MQ for VSE V3.0, there are no immediate plans to withdraw support for this SupportPac within the same timeframe.

Installation Instructions:

Download file mqc5.zip into a temporary directory, and unzip it. This will create the necessary library files and readme text file. Follow the steps in the Installation section of the readme file for details of how to transfer the files to VSE and to set up the SupportPac for your environment.

Unzipping the MQC5.ZIP file should create or replace the following files:

• MQICLIB.JCL	- catalogs MQ Client objects and phases.
• MQICINC.JCL	- catalogs copybooks and include files
• MQICSMP.JCL	- catalogs example client application sources
• MQICBRG.JCL	- catalogs components of the MQ Client for VSE Bridge
• README.TXT	- please read this file carefully.
• LICENSES*.TXT	

Upload the .JCL files to the z/VSE reader (or to z/VM) in BINARY and LRECL 80. These files catalog PHASEs, OBJs, include files, copy books and examples into a library of your choice. There is a PAUSE statement in the jobs that allow you to specify the destination sublibrary via a SETPARM statement. The default sublibrary is PRD2.MQICVSE. MQICLIB.JCL should be run as first, since it creates the destination sublibrary if it does not already exist.

Note: Do not install the MQ Client into the same sublibrary where MQ Server is installed.

LIBDEF considerations:

You should add the MQ Client installation sublibrary to the PHASE LIBDEF chain of your applications as well as CICS. The MQ Client code will load MQ Client specific phases during execution of MQ calls. If these MQ Client phases are not available, applications will abend when calling an MQ Client function.

When compiling MQ Client applications, you need to have the MQ Client installation sublibrary in the SOURCE LIBDEF in your compile jobs. You should have only the MQ Client sublibrary, but not the MQ Server sublibrary included. This ensures that the include files and copybooks from the MQ client are used. The declarations in the MQ Client include files and copybooks should be compatible with those from the MQ server. The MQ client include files and copybooks contain some additional definitions needed for the MQ client only (e.g. structures for MQCONNX call).

CICS Definitions:

You must define the MQ Client programs “MQICVSEP”, “AMQERROR” and “UAMQMSET” to CICS, unless you have program autoinstall active:

```
DEFINE PROGRAM(MQICVSEP) GROUP(MQM) LANGUAGE(C)
       EXECKEY(CICS) DATALOCATION(ANY)
DEFINE PROGRAM(AMQERROR) GROUP(MQM) LANGUAGE(ASM)
       EXECKEY(USER) DATALOCATION(ANY)
DEFINE PROGRAM(UAMQMSET) GROUP(MQM) LANGUAGE(ASM)
       EXECKEY(USER) DATALOCATION(ANY)
```

Any CICS program that uses the MQ Client must also be defined to CICS with **EXECKEY(CICS)**. This is because the MQ Client is called using a branch and link interface, where CICS cannot perform a key switch. So the calling program must already run in CICS key.

If you plan to use the MQ Client for VSE Bridge, then you should also perform CICS CSD file updates via member MQCICSD.Z.

Application changes to specify the MQ Server to work with

When a z/VSE application is using MQ Server on z/VSE, it does not need to know with which MQ Server it is working, since there is only one MQ Server running (i.e. one per CICS region). Thus applications always connect to their local MQ Server.

When using MQ Client, then the application must tell the MQ Client with which MQ Server it wants to work with, and where the MQ Server is located (i.e. IP address or hostname).

There are 2 options how an application specifies the MQ Server it wants to work with:

- **MQCONN with MQSERVER environment variable:** While the MQCONN call itself remains unchanged, the application must set the MQSERVER environment variable prior to calling MQCONN. Setting environment variables can be done via the setenv() call when using LE/C. COBOL and PL/1 applications must use the MQSETENV API call, which is provided by the MQ Client for VSE. Environment variables can also be set in the LE runtime default options (see CEEUOPT, CEEDOPT and CEECOPT).

- **MQCONNX:** This requires changing the MQCONN call to an MQCONNX call and setup the MQCNO and MQCD data structures accordingly.

In both cases the client-connection channel name must be specified like

```
ChannelName/TransportType/ConnectionName
```

or

```
ChannelName/TransportType/ConnectionName(PortNumber)
```

where ChannelName is the name of the server-connection channel that has been defined on the MQ Server. TransportType must be TCP, and ConnectionName specifies the IP address or hostname of the MQ Server. If the MQ Server uses a different port number than the default port (1414), then you must also specify the port number.

The corresponding server-connection channel must be defined in the MQ Server, unless you are using automatic defined channels.

Considerations for DEVELOPMENT, TEST and PRODUCTION systems:

In an MQ Client environment the application and the MQ server runs on different systems. Typically a VSE customer has different environments. There is a DEVELOPMENT environment, a TEST environment, a PRODUCTION environment, etc. Each environment has separate VSE systems and separate MQ Servers as well.

In an MQ Server environment, the application always uses to the local MQ server, which is running on the same VSE system as the applications. Thus, the application is automatically connecting to its MQ Server, regardless on which environment it runs.

An MQ Client application running in the TEST environment needs to connect to the TEST MQ Server; once running in the PRODUCTION environment, it then needs to connect to the PRODUCTION MQ server instead. To accomplish this, the application needs some kind of logic to 'know' in which environment it currently runs, and pick the appropriate MQ server.

Application changes to handle MQ Client specific errors

Because MQ client applications use a remote MQ Server, additional error situations can occur. Each MQ Client MQI call requires network communication through the MQ Client Channel over TCP/IP. Network connections may break or get interrupted for various reasons (firewalls idle timeouts, etc).

The MQ Client application must be able to react appropriately on such error situations. There may be additional error codes from MQI calls, and you may need to implement more complicated recovery code in the applications.

Application changes in regards of data translation (ASCII/EBCDIC)

When a z/VSE application works with an MQ Server on z/VSE, the data that is contained in MQ message is usually in a host format (e.g. EBCDIC for textual data). When passing messages from an MQ Server on z/VSE to a non-z/VSE MQ Server, the messages can be converted automatically when transmitted via the server-to-server channel.

When using MQ Client, the applications directly work with an MQ Server that is located outside of z/VSE, and thus mostly on an ASCII-based platform (e.g. Linux, Windows, Unix). Because of that, care must be taken about the data format of the messages passed through MQ Client to other systems. Messages are not automatically converted, but the MQ Client application can request to convert the message with the MQGET call by using option MQGMO-CONVERT. This performs message conversion for known message formats.

To specify the character set used by the MQ Client application use the environment variable MQCCSID. Setting environment variables can be done via the setenv() call when using LE/C. COBOL and PL/1 applications must use the MQSETENV API call, which is provided by the MQ Client for VSE. Environment variables can also be set in the LE runtime default options (see CEEUOPT and CEECOPT).

Note: When using an MQ Client, codepage conversion (ASCII/EBCDIC) is performed by the MQ Server.

Re-link of the applications against MQ Client library

You must re-link the applications against the MQ Client library (MQICVSE.OBJ).

When linking applications against the MQ Server library, each MQI call will cause the appropriate object deck to be included automatically. For example the MQGET call will cause MQGET.OBJ to be included via autolink.

When using the MQ Client, you must explicitly include the MQ Client object deck MQICVSE.OBJ to your application. MQICVSE contains the entries of all MQI calls (e.g. MQGET). Make sure that the MQ Client sublibray is in the LIBDEF of your compile and link jobs.

In addition, applications using the MQ Client must be prelinked using the LE prelinker. Prelinking is usually only required for applications written in the LE/C programming language. However, when using the MQ Client, you must prelink all your applications, even when they are implemented in COBOL or PL/1.

```
// OPTION CATAL
   PHASE YOURPROG,*
   INCLUDE YOURPROG
   INCLUDE MQICVSE
/*
// EXEC EDCPRLK,SIZE=EDCPRLK,PARM='UPCASE MAP'
/*
// EXEC LNKEDT,SIZE=256K
/*
```

MQ Server features not supported by MQ Client

The following features that are supported by WebSphere MQ Server for z/VSE V3.0.0 (5655-U97) are not supported or not available by the MQ Client for z/VSE:

Publish and Subscribe (Pub/Sub) is not supported

WebSphere MQ Server for VSE supports Publish and Subscribe since APARs PM85374 and PM86869. These added the following MQI calls to the MQ Server on z/VSE:

• MQSUB	Register subscription
• MQSUBRQ	Subscription request

These MQI calls are not available with the MQ Client on z/VSE.

The lack of the MQSUB and MQSUBRQ calls makes it impossible to use publish and subscribe features from a z/VSE application with MQ Client. Existing VSE MQ Server applications using these features will fail when using MQ Client instead and will most likely need to be redesigned.

Message property API calls are not supported

WebSphere MQ Server for VSE supports Message Property API calls since APARs PM85374 and PM86869. These added the following MQI calls to the MQ Server on z/VSE:

• MQBUFMH	Convert buffer into message handle
• MQCRTMH	Create message handle
• MQDLTMH	Delete message handle
• MQDLTMP	Delete message property
• MQINQMP	Inquire message property
• MQMHBUF	Convert message handle into buffer
• MQSETMP	Set message property

These MQI calls are not available with the MQ Client on z/VSE.

The lack of the Message Property calls makes it impossible to use message property features from a z/VSE application with MQ Client. Existing VSE MQ Server applications using these features will fail when using MQ Client instead and will most likely need to be redesigned.

The WebSphere MQ Client for VSE does not support SSL enabled channels.

While the MQ Server for VSE supports SSL enabled channels, the MQ Client does not support SSL. Thus, all data is transported ‘in clear’ over the Server Channel to the MQ Server.

Some VSE customers have requirements to encrypt all data that leaves the VSE system. This requirement could not be fulfilled when using MQ Client on VSE. Even when a VSE application uses an MQ Server running on Linux on System z on the same box, there still is the possibility that data transported in clear is leaked.

DB2 Q Capture requires an MQ Server environment on z/VSE

The DB2 DataPropagator Q Capture Supplement for DB2 on VSE & VM requires an MQ Server environment on z/VSE to run the Q Capture program.

Thus, the DB2 DataPropagator Q Capture feature can not be used in a MQ client only environment.

MQ CICS Bridge is not available with the MQ Client

The WebSphere MQ for VSE Server function MQ CICS Bridge is not available with the MQ Client.

Thus, the MQ CICS Bridge feature cannot be used in a MQ client only environment.

The MQ Client does not support higher versions of various MQ structures

Many of the structures used with the MQ programming interface contain a version field. With this, additional fields can be added to the structure at the end. The MQ Client does not support higher version numbers of various MQ structures, and thus does not support various features which are related to newer versions of the structures. Most structures only support Version 1 (e.g. MQOD, MQPMO, MQMD, MQGMO, MQCFH), except MQCD which supports up to Version 3.

Chapter 3: Using the z/VSE MQ Client Trigger Monitor

Triggering is a must-have feature when asynchronous message passing is used. It allows to start (trigger) an application when one or more messages arrive on a queue. Per queue, you can define which trigger program is to be triggered. When invoked (triggered), the trigger program gets information about the trigger event it was invoked for. Besides others, the trigger event contains information about the queue that caused the trigger. The triggered program then performs MQGET operations to get the messages from the queue and processes them.

Unlike MQ Client packages available for Linux/Unix/Windows or even z/OS, the z/VSE MQ Client package does not contain a client trigger monitor that would allow triggering an application when a message arrives on a queue.

On distributed MQ Client installations you would run the MQ Client Trigger Monitor program RUNMQTMC to let it connect to an MQ Server and monitor an initiation queue. On z/OS the trigger monitor transaction CKTI can be used.

Neither RUNMQTMC nor CKTI is available with the z/VSE MQ Client.

z/VSE MQ Client Trigger Monitor

The z/VSE MQ Client Trigger Monitor function provided with z/VSE V6.1, APAR PIPI42612 / PTF UI28408 (z/VSE 5.1), or APAR PI42615 / PTFUI28409 (z/VSE 5.2) implements such a trigger monitor to be used with the MQ Client on z/VSE. Similar to the MQ Server on z/VSE, the z/VSE MQ Client Trigger Monitor runs under CICS and allows triggering CICS applications when messages arrive on queues.

The triggering performed by the z/VSE MQ Client Trigger Monitor works the same way as with WebSphere MQ for z/VSE V3.0. Thus no changes in the triggered applications are required specific to the use of the z/VSE MQ Client Trigger Monitor. However, the triggered applications must have been migrated to use the MQ Client instead of the MQ Server as described in the previous chapter.

The z/VSE MQ Client Trigger Monitor works exactly the same way as trigger monitors on other platforms. Thus the trigger setup and configuration is the same as for other triggering monitors.

How triggering works on MQ Servers

There is a nice description of how triggering works in the following blog entry "Triggering for beginners":
https://www.ibm.com/developerworks/community/blogs/aimsupport/entry/triggering_for_beginners?lang=en

The following section describes the configurations settings that enable and control triggering.

Configuring Triggers:

Triggering can be configured for a local queue on an MQ Server. To enable triggering you configure the following trigger settings on the queue:

- **Trigger type:** Can be one of the following:
  - **FIRST:** A trigger event occurs when the current depth of the triggered queue changes from 0 to 1. Use this type of trigger when the serving program will process all the messages on the queue
  - **EVERY:** A trigger event occurs every time a message arrives on the triggered queue. Use this type of trigger when the serving program will only process one message at a time.
  - **DEPTH:** A trigger event occurs when the number of messages on the triggered queue reaches the value of the Trigger Depth attribute. Use this type of trigger when the serving program is designed to process a fixed number of messages.

- **Trigger Depth:** Used for a trigger type of DEPTH, see above.

- **Trigger Data:** Data passed with the trigger event to the triggered application. For MQ Server on VSE as well as for the z/VSE MQ Client Trigger Monitor this is a 13 character field containing
  - 4 character CICS transaction ID or 4 blanks or underscores
  - 8 character CICS program name or 8 blanks or underscores
  - 1 character trigger event flag
    - 'F': Trigger type is FIRST
    - 'E': Trigger type is EVERY
    - Applications can set this to 'S' to stop the trigger monitoring.

- **Initiation Queue**: The name of the initiation queue. When the criteria for a trigger event are met, the queue manager puts a trigger message on the initiation queue. Create your own local initiation queue, or use the SYSTEM.DEFAULT.INITIATION.QUEUE.

- **Process Name:** The name of a WebSphere MQ process definition. For the z/VSE MQ Client Trigger Monitor this field is not really used, but MQ Servers requires that a valid process name is specified, otherwise triggering is not activated. You can define a dummy process (e.g. named CICS.PROGRAM) with the following attributes:
  - **Process name:** CICS.PROGRAM
  - **Application Type:** CICS VSE (it is not really relevant what you choose here)
  - **Application ID:** CICSPROG (it is not really relevant what you specify here)
  - **Environment Data:** a free form text field that is passed with the trigger event to the triggered program.
  - **User Data:** a free form text field that is passed with the trigger event to the triggered program.

Trigger processing:

The following processing is performed by an MQ Server when a message arrives on a queue which has triggering enabled.

- When the trigger criteria is met (dependent on trigger type), the MQ Server creates a trigger event message and puts it onto the initiation queue.
  - The trigger event message contains information about the queue that the trigger event is created for, the process name (e.g. CICS.PROGRAM), the trigger data field (13 characters for use with MQ Server on VSE as well as for the z/VSE MQ Client Trigger Monitor, see above), and the information from the associated process definitions, such as application type and ID, environment data and user data.
  - Please see structure MQTM for more details.

Trigger Monitors:

- A trigger monitor (usually started as separate process) monitors the initiation queue, and gets the trigger event message from the initiation queue.
  - Please note that a trigger monitor may run on the same system as the MQ Server (Server Trigger Monitor), or it may run on a different system, connecting to the MQ Server using an MQ client (Client Trigger monitor).
- Based on the information in the trigger event message, the trigger monitor starts the trigger program (or process).
  - On z/VSE the trigger program is started via EXEC CICS LINK or via EXEC CICS START of the trigger transaction, as specified in the trigger data field (see above). The triggered program can retrieve the complete

trigger event message (MQTM structure) either through the COMMAREA (for LINK) or via EXEC CICS RETRIEVE (for START).

- The triggered program connects to the queue manager (e.g. via MQ Client), opens the queue that caused the trigger (specified in field QName in the MQTM structure), and performs MQGET operations to retrieve the application message that has initiated the trigger.
    - Triggered programs can also use information from the other fields of the MQTM structure, such as the user data or the environment data fields.
- Typically a triggered application performs MQGET (with a small wait time) in a loop, until no more messages are available on the queue. This ensures that it also handles application messages that arrived on the queue in the timeframe where the trigger event was created, and the application started to process messages.

Installing the z/VSE MQ Client Trigger Monitor

The z/VSE MQ Client Trigger Monitor is preinstalled on z/VSE V6.1, or is installed either by applying APAR PI42612 / PTF UI28408 (z/VSE 5.1), or APAR PI42615 / PTF UI28409 (z/VSE 5.2).

The z/VSE MQ Client Trigger Monitor consist of just one CICS program named IESMQCTM.

Besides cataloging the program (IESMQCTM.PHASE) into PRD1.BASE, the following CICS definitions must be performed:

- Define program IESMQCTM to CICS:
    - Program IESMQCTM
    - Group: VSESPG
    - Language: C
    - ExecKey: CICS
    - DataLocation: Any

- Define transaction MQTM to CICS:
    - Transaction: MQTM
    - Group: VSESPG
    - Program: IESMQCTM
    - TaskDataKey: User
    - TaskdataLoc: Any
    - Profile: IESXACT1

You also must define transaction MQTM to your security manager. For the z/VSE basic security manager (BSM) perform the following commands with BSTADMIN:

```
ADD TCICSTRN 'MQTM' UACC(NONE) DATA('IBM SUPPLIED')
```

```
      PERMIT TCICSTRN 'MQTM' ID(GROUP01) ACCESS(READ)
      PERFORM DATASPACE REFRESH
```

A job is supplied with the installation package that performs above definitions (ftp://public.dhe.ibm.com/eserver/zseries/zos/vse/download/mqtmdefs.job).

The z/VSE MQ Client Trigger Monitor issues a couple of new messages (prefixed with IESC60). You can load the message explanation of these new messages into the z/VSE message explanation online (OME) file using the supplied OME-update job (ftp://public.dhe.ibm.com/eserver/zseries/zos/vse/download/mqtmome.job).

Both, the CICS definitions and the OME messages are already provided by default in z/VSE 6.1.

Configuring the z/VSE MQ Client Trigger Monitor

Once the z/VSE MQ Client Trigger Monitor has been installed, you need to configure it.

You can start several instances of the z/VSE MQ Client Trigger Monitor at the same time. One trigger monitor instance monitors a single initiation queue on an MQ Server. To monitor different initiation queues on the same or on different MQ servers, you need to start multiple instances of the z/VSE MQ Client Trigger Monitor, one for each initiation queue to monitor.

The user interface to the z/VSE MQ Client Trigger Monitor is provided by the MQTM transaction, that you can use from a terminal, or from the z/VSE console (via MSG F2,DATA=MQTM …).

The general syntax used with the MQTM transaction is

```
      MQTM command parameter1=val1,parameter2=val2,...
```

The parameters may be optional per command. There must be at least one blank between the command and the first parameter. Several parameters are separated by a comma. Blanks are allowed before and after the comma, but they do not belong to the parameter name or value. Parameters are usually in “key=value” format.

The following sections describe the available commands in detail.

Usually the MQTM transaction is entered interactively by a user. However, you can also use the MQTM transaction from a program, to automate certain actions with MQTM. To do so, you can use EXEC CICS START to start the MQTM transaction and pass the command string (command name plus parameters) as data. Alternatively, you can perform an EXEC CICS LINK to program IESMQCTM and pass the command string via COMMAREA.

Start a trigger monitor instance:

To start a trigger monitor instance, use the following START command:

```
MQTM START MQSERVER=server-channel-name/TCP/ip-or-hostname
      [,QMGR=queue-manager-name]
      [,INITQ=initiation-queue-name]
      [,RECONTIME=n]
      [,RECONTRIES=n]
      [,WAITTIME=n]
```

The **MQSERVER** parameter is required. It specifies the MQ Server to connect to. The format is "`server-channel-name/TCP/ip-or-hostname`" (like for the MQSERVER environment variable used by MQ Client applications).

The **QMGR** parameter is optional. It specifies the name of the queue manager on the MQ Server. If omitted, the default queue manager on the specified MQ Server is used.

The **INITQ** parameter is optional. It specifies the name of the initiation queue used for triggering. If omitted, the system default initiation queue SYSTEM.DEFAULT.INITIATION.QUEUE is used.

The **RECONTIME** parameter is optional. It specifies the time in seconds after which the trigger monitor tries to reconnect after an existing connection to the MQ Server was interrupted. If RECONTIME is omitted, then the default of 30 seconds is used. If RECONTIME specifies 0 (zero), then the trigger monitor does not try to reconnect at all and will terminate when the connection is interrupted.

The **RECONTRIES** parameter is optional. It specifies the number of times the trigger monitor tries to reconnect after an existing connection to the MQ Server was interrupted. If RECONTRIES is omitted, or RECONTRIES specifies 0 (zero), then it will try infinitely.

The **WAITTIME** parameter is optional. It specifies the time-out in seconds that the trigger monitor uses with the MQGET call against the initiation queue. If WAITTIME is omitted, then the default of 60 seconds is used. If WAITTIME specifies 0 (zero), then MQGET will wait infinitely. The use of a time-out with MQGET helps to keep the client connection open, since idling connections may get interrupted by firewalls.

Stop a trigger monitor instance:

To stop an active trigger monitor instance, use the following STOP command and specify the name of the MQ Server, queue manager and initiator queue to identify the instance to stop.

```
MQTM STOP MQSERVER=server-channel-name/TCP/ip-or-hostname
     [,QMGR=queue-manager-name]
     [,INITQ=initiation-queue-name]
```

The **MQSERVER** parameter is required. It specifies the MQ Server to connect to. The format is "`server-channel-name/TCP/ip-or-hostname`" (like for the MQSERVER environment variable used by MQ Client applications).

The **QMGR** parameter is optional. It specifies the name of the queue manager on the MQ Server. If omitted, the default queue manager on the specified MQ Server is used.

The **INITQ** parameter is optional. It specifies the same of the initiation queue used for triggering. If omitted, the system default initiation queue SYSTEM.DEFAULT.INITIATION.QUEUE is used.

Display a list of all active trigger monitors:

To display a list of all active trigger monitor instances, use the SHOW command. When the command is issued from the console (e.g. MSG F2,DATA=MQTM SHOW), then the list is printed to the console. If the command is issued from a terminal, then a 3270 screen will be displayed on the terminal. You can scroll through the screens, if more instances are active than fit on one screen. You can refresh the display by pressing the ENTER key.

```
MQTM SHOW
```

Start all pre-configured trigger monitor instances:

In an environment with multiple trigger monitor instances, you typically do not start them manually (via MQTM START). Instead, you can provide a list of trigger monitors to start and start them all at once.

The STARTALL command will read the start up list library member IESMQCTL.Z which is searched for in the active LIBDEF SOURCE chain. It is a good practice to place IESMQCTL.Z into PRD2.CONFIG.

```
MQTM STARTALL
```

Member IESMQCTL.Z is plain text and contains START commands for those trigger monitor instances that are to be started. START commands must begin with the 'START' keyword, followed by parameters such as MQSERVER, QMGR, INITQ and RECONTIME, etc. To continue the command on the next line place a '-' character at the end of the line. This indicates that the command is continued at the next line. Lines beginning with an asterisk ('*') are treated as comments and are ignored. Empty lines (all blanks) are also ignored.

Example:

```
*
START MQSERVER=server-channel-name/TCP/ip-or-hostname -
[,QMGR=queue-manager-name] -
[,INITQ=initiation-queue-name] -
[,RECONTIME=n] -
[,RECONTRIES=n] -
[,WAITTIME=n]
*
```

Stop all active trigger monitor instances:

The STOPALL command stops all currently active trigger monitor instances.

```
MQTM STOPALL
```

Automatic start-up of trigger monitors during CICS start-up:

If you wish to start up the pre-configured trigger monitor instances when CICS is started up, then you can include program IESMQCTM into the DFHPLTPI list of CICS. This will perform the same action as a STARTALL command and will also read the start up list library member IESMQCTL.Z.

```
DFHPLT TYPE=ENTRY,PROGRAM=IESMQCTM
```

Automatic shutdown of trigger monitors during CICS shutdown:

If you wish to shut down all active trigger monitor instances when CICS is terminated, then you can include program IESMQCTM into the DFHPLTSD list of CICS. This will perform the same action as a STOPALL command.

```
DFHPLT TYPE=ENTRY,PROGRAM=IESMQCTM
```

Problem determination and tracing

When a trigger monitor does not work as expected, the following can help to find the cause of the problem:

1. Several messages issued by the z/VSE MQ Client Trigger Monitor contain MQ return and reason codes. Please use the MQ documentation and/or the copybooks provided with MQ Client to determine the meaning of these return and reason codes.

2. You can turn on tracing for the z/VSE MQ Client Trigger Monitor. Several trace messages will then be issued either to SYSLOG, SYSLST or both. To turn on tracing, add the TRACE=nnn parameter to the MQTM command. The TRACE parameter can have the following values:
    - TRACE=SYSLST
    - TRACE=SYSLOG
    - TRACE=BOTH
    - TRACE=OFF (default)

   To trace the activities of a trigger monitor instance the TRACE parameter must have been specified at the START or STARTALL command. You can not dynamically turn on tracing for an already active instance. You would need to stop and restart it with the TRACE parameter specified in this case.

Remarks

Trademarks

Comments and Questions

Comments or questions on this documentation are welcome. Please send your comments to:

zvse@de.ibm.com