z/TPF QUDM Driver

User's Guide

ZTEST QUDM

The purpose of the QUDM driver is to send user-defined metrics (UDM) through the **tpf_rtmc_send_user_defined_metrics** function.

The ECBs for the QUDM driver run with a name-value pair. The name is MsgType and the value is IQUDMDriver.

There are two types of user-defined metrics that can be sent with the QUDM driver: sawtooth data and data that is already in JSON format.

Sawtooth Data

Sawtooth data is sent with a datatype of QUDM.
Sawtooth data is always sent in continuous mode and is balanced across I-streams. That is, the START parameter must be specified to send sawtooth data. An ECB is created with CRETC every second to send the data continuously.
A 3-entry array of data is created. Each array entry contains 4 elements: a group name, a millisecond value, a microsecond value and a fixed value.

- The first entry in the array has a group name of mils, a millisecond value based on the current timestamp, a microsecond value based on the current timestamp and a fixed value of 42.
- The second entry in the array has a group name of plus1000, a millisecond value based on the current timestamp plus 1000, a microsecond value based on the current timestamp plus 1000 and a fixed value of 777.
- The third entry in the array has a group name of plus2000, a millisecond value based on the current timestamp plus 2000, a microsecond value based on the current timestamp plus 2000 and a fixed value of 1337.

Because of the increase in values of each element, the data can be thought of as a sawtooth.

When starting the QUDM driver to send sawtooth data, you specify the number of times to send the sawtooth data per second (the rate).

Many instances of the QUDM driver can be started to send sawtooth data. You specify a unique token on each START command to identify the instance of the driver. You can specify a different rate on each instance of the driver.
There is currently no display for the driver to get information on the different instances (this would be a good future enhancement).
The instances of the driver are kept in a table in system heap. The tables are subsystem unique.
The best way to see what instances of the driver, if any, are running are to enter these commands (depending on the subsystem you are on):
ZDSHP ‘IQUDMBSS’
ZDSHP ‘IQUDMWP ’ (trailing blank)

Then ZDCOR the system heap address.
For example, let’s say one instance of the driver has been started on the BSS subsystem with a token of 001 and a rate of 100 messages per second. An entry in the system heap table is mapped by struct TPF_QUDM_ENTRY in header qudm.hpp. Each entry contains:

- The token.
- A status indicator. Y indicates the instance is started, N indicates the instance is stopped.
- The number of ECBs running this instance of the driver.
- The rate.

The following display indicates there is one instance started, running at hex 64 = 100 messages per second, with token 001 and there is 1 ECB running that instance:


```
ZDSHP 'BSS QUDM'
CSMP0097I 16.08.44 CPU-B SS-BSS  SSU-HPN  IS-01
DSHP0040I 16.08.44 SYSTEM HEAP UNIQUE TOKEN INFORMATION DISPLAY
 TOKEN NAME    BSS QUDM
 HEX TOKEN     C2E2E240D8E4C4D4 0000000000000000 0000000000000000
               0000000000000000 0000000000000000 0000000000000000
               0000000000000000 0000000000000000
 OWNER NAME    drvrQUDM BSS
 MEMORY ALLOCATION
  ADDRESS      000000117842E000          SIZE  4K
  FRAMES       0000000000000001          TYPE  4
  LOCATION     64 BIT SYSTEM HEAP        INDEX 000
  ALLOCATED AT 06/13/2023 10.48.10

END OF DISPLAY

ZDCOR 000000117842E000.FFF
CSMP0097I 16.08.49 CPU-B SS-BSS  SSU-HPN  IS-01
DCOR0010I 16.08.49 BEGIN DISPLAY
 000000117842E000- D8E4C4D4 E3C1C2D3 C5404040 00000001 QUDMTABL E   ....
 000000117842E010- F0F0F1E8 00000001 00000064 00000000 001Y.... ........
 000000117842EFF0- 00000000 00000000 00000000 000000   ........ .......

END OF DISPLAY - ZEROED LINES NOT DISPLAYED
```

The command to start sending sawtooth data is:
ZTEST QUDM START RATE-xxxx TOKEN-yyy

The command to stop sending sawtooth data is:
ZTEST QUDM STOP TOKEN-yyy|ALL

When running the QUDM driver in continuous mode (sending sawtooth data), certain error return codes from the **tpf_rtmc_send_user_defined_metrics** function are ignored and the driver continues to attempt to send data since these errors might be temporary. These error return codes are ignored:

- TPF_RTMC_UDM_NOT_ACTIVE
- TPF_RTMC_UDM_SEND_ERROR
- TPF_RTMC_UDM_NOT_IN_CORRECT_STATE
- TPF_RTMC_UDM_TIMEOUT

Data that is in JSON format

JSON format data is sent with any data type other than QUDM or QMWR. You specify the data type.
JSON format data is not sent in continuous mode.
You specify the name of the file that contains the data that is in JSON format. You also specify how many times to include the data in the file before the total data is sent.
Sending data that is in JSON format is a good way to test sending a large amount of user-defined data.

The command to send data that is already in JSON format is:
ZTEST QUDM JSON DATATYPE-type FILE-jsonfile PACKAGES-n

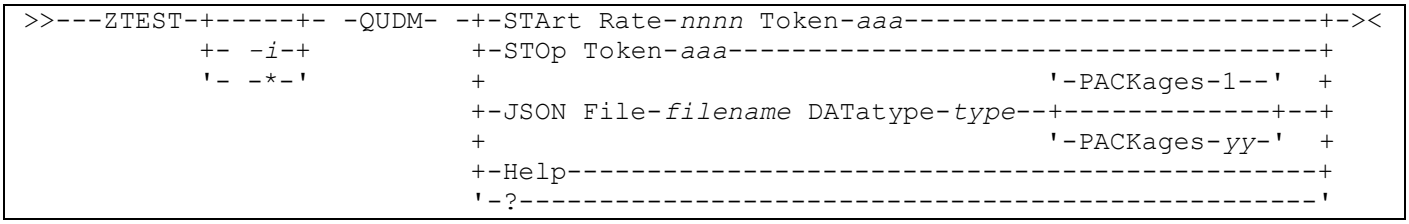
When running the QUDM driver to send data that is in JSON format, no error return codes from the **tpf_rtmc_send_user_defined_metrics** function are ignored.

Requirements and restrictions

- The ZRTMC command with the START and UDM parameters specified must be entered to start collecting user-defined metrics data.

- A token of ALL (any case) is not allowed on the START function of the driver. It is a reserved parameter for the STOP function of the driver.
- When running the QUDM driver to send data that is in JSON format, a datatype other than QUDM or QMWR must be specified.
- When running the QUDM driver to send data that is in JSON format, the file containing the data must contain data array entries. Each entry must start with '{' and end with '}'. For example:
  {"groupName":"name1","metric1":51,"metric2":298,"metric3":402}
  There can be one entry or many entries. The entries must be comma separated.

Format



i

indicates the specific I-stream in which the driver will be run. If *i* is not specified, the test case(s) will be executed on the I-stream on which the command is entered. Note that when running in continuous mode, the driver is balanced across all I-streams.

*****

specifies the driver will be invoked on all currently defined and available I-streams.

STArt

specifies to start sending sawtooth data in continuous mode.

Rate-*nnnn*

specifies the number of times to send the sawtooth data per second, where *nnnn* is a 1- to 4-digit decimal number.

Token-*aaa*

specifies the token to assign to this instance of the driver, where *aaa* is a unique 3-character alphanumeric token.
When the **START** parameter is specified, a token of **ALL** (any case) is not allowed.
When the **STOP** parameter is specified, a token of **ALL** indicates all instances of the driver are stopped.

STOp

specifies to stop sending sawtooth data.

JSON

specifies to send data that is already in JSON format.

File-*filename*

specifies the file containing the data in JSON format, where *filename* is the 1- to 1023-character name and location of the file.

DATatype-*type*

specifies the data type to assign to the data, where *type* is the 1- to 64-character alphanumeric data type. A data type of QUDM or QMWR is not allowed. These are used for other types of data.

PACKages-*yy*

specifies how many times to include the data in the file before the total data is sent, where *yy* is a 1- to 7-digit decimal number.

Help | ?

displays the correct syntax of the command.

Examples

The following example starts an iteration of the driver to send continuous sawtooth data at a rate of 100 times per second. The token assigned to the iteration is 001:

```
User:   ZTEST QUDM START TOKEN-001 RATE-100

System: QUDM0025I 16.08.40 QUDM DRIVER STARTED WITH TOKEN 001.
```

The following example stops the iteration of the driver assigned a token value of 001:

```
User:   ZTEST QUDM STOP TOKEN-001

System: QUDM0029I 17.03.51 QUDM DRIVER WITH TOKEN 001 STOPPED.
```

The following example stops all iterations of the driver:

```
User:   ZTEST QUDM STOP TOKEN-ALL

System: QUDM0032I 11.50.00 ALL INSTANCES OF THE QUDM DRIVER ARE STOPPED.
```

The following example sends data that is already in JSON format. The JSON data is in the file at location /tmp/jsondata.json. A data type of DATA1 is assigned to the data. The data in the file is repeated 100000 times before the data is sent:

```
User:   ZTEST QUDM JSON FILE-/tmp/jsondata.json DATATYPE-DATA1 PACKAGES-100000

System: QUDM0015I 09.22.03 THE JSON FILE SIZE IS 66, THE TOTAL PAYLOAD SIZE IS
        6700112.
        QUDM0022I 09.22.04 THE JSON PAYLOAD WAS SENT SUCCESSFULLY.
```

Messages

Below is a list of the QUDM driver messages:

QUDM0001E RTRTMC IS NOT PROCESSING UDM DATA.

Explanation: Real-time runtime metrics collection is not collecting user-defined metrics data.

System action: The z/TPF system ends the QUDM driver.

User response: Complete the following actions:

1. Enter the ZRTMC command with the START and UDM parameters specified to start collecting user-defined metrics data.
2. Start the QUDM driver again.

QUDM0002E THE ENDPOINT GROUP NAME IS NOT VALID.

Explanation: The ZRTMC command was entered with the MODIFY and UDM parameters specified for a specific endpoint group name, but the name is not valid.

System action: The z/TPF system ends the QUDM driver.

User response: Complete the following actions:

1. Enter the ZRTMC command with the STOP and UDM parameters specified to stop collecting user-defined metrics data.
2. Enter the ZRTMC command with the MODIFY, UDM and EPT parameters specified to specify a correct endpoint group name.
3. Enter the ZRTMC command with the START and UDM parameters specified to start collecting user-defined metrics data.
4. Start the QUDM driver again.

QUDM0003E THE INPUT IS NOT VALID. THE POINTER PASSED IS NULL.

Explanation: The pointer to the parameters passed to the tpf_rtmc_send_user_defined_metrics function is NULL.

System action: The z/TPF system ends the QUDM driver.

User response: This error should not occur. Report the error to the driver owner.

QUDM0004E AN ERROR OCCURRED SENDING THE DATA.

Explanation: A high-speed connector error occurred when sending the user-defined metrics data.

System action: The z/TPF system ends the QUDM driver.

User response: Determine the cause of the high-speed connector error.

QUDM0005E THE SYSTEM IS NOT IN THE CORRECT STATE.

Explanation: The z/TPF system is not in a state where data can be sent through the high-speed connector.

System action: The z/TPF system ends the QUDM driver.

User response: Cycle the z/TPF system to a higher system state.

QUDM0006E A TIMEOUT OCCURRED WHILE SENDING THE DATA.

Explanation: The data could not be sent to the endpoint group that was defined for user-defined metrics. Network issues can cause packet loss and timeouts when the data is being sent.

System action: The z/TPF system ends the QUDM driver.

User response: Investigate whether there is a network issue.

QUDM0007E THE METRICS EXCEED THE MAXIMUM SIZE.

Explanation: The user-defined metrics data exceeds the high-speed connector limit of 1 MB. On IBM® z15™ servers or later, the data is automatically compressed.

System action: The z/TPF system ends the QUDM driver.

User response: Take one of the following actions:
- Reduce the amount of data in the JSON file specified.
- Decrease the number of packages specified.

QUDM0008E THE METRICS DATA IS EMPTY.

Explanation: No data was passed to the tpf_rtmc_send_user_defined_metrics function.

System action: The z/TPF system ends the QUDM driver.

User response: This error should not occur. Report the error to the driver owner.

QUDM0009E INSUFFICIENT MALLOC STORAGE IS AVAILABLE.

Explanation: Real-time runtime metrics collection could not allocate 64-bit ECB heap to send user-define metrics data.

System action: The z/TPF system ends the QUDM driver.

User response:
Do one of the following:

- Reduce the size of the data being sent:
  - Reduce the amount of data in the JSON file specified.
  - Decrease the number of packages specified.
- Review your 1 MB frame usage.
- Enter the ZCTKA ALTER command with the XMMES parameter specified to change the maximum number of 1 MB frames that an ECB can acquire for 64-bit ECB heap, if necessary.

Enter the ZTEST QUDM command again.

QUDM0010E A DFDL ERROR OCCURRED WHILE CONVERTING THE STRUCTURE TO JSON.

Explanation: A DFDL error occurred while the tpf_rtmc_send_user_defined_metrics function was preparing the data to send.

System action: The z/TPF system ends the QUDM driver.

User response: This error should not occur. Report the error to the driver owner.

QUDM0011E THE udm_data JSON NODE WAS NOT FOUND.

Explanation: The udm_data JSON node was not included in the data passed to the tpf_rtmc_send_user_defined_metrics function.

System action: The z/TPF system ends the QUDM driver.

User response: This error should not occur. Report the error to the driver owner.

QUDM0012E UNEXPECTED RETURN CODE RECEIVED FROM tpf_rtmc_send_user_defined_metrics RETURN CODE *nn*

Where:
nn the return code returned by the function

Explanation: The tpf_rtmc_send_user_defined_metrics function returned a return code that was not expected.

System action: The z/TPF system ends the QUDM driver.

User response: This error should not occur. Report the error to the driver owner.

QUDM0013E QUDM DRIVER NOT STARTED. UNABLE TO OBTAIN SYSTEM HEAP.

Explanation: The QUDM driver could not allocate system heap for an internal table.

System action: The command is rejected.

User response:

Complete the following steps:

1. Enter ZSTAT SYSHEAP HIGH-0 to display system heap storage statistics and the top users of 64-bit system heap.
2. If you do not have enough 64-bit system heap, complete the following steps to increase the size of your 64-bit system heap:
    a. Take one of the following actions:

- Enter the ZCTKA ALTER command with the SHA parameter specified to increase the size of the preallocated 64-bit system heap area.
- Enter the ZCTKA ALTER command with the FRM1MB parameter specified to increase the number of 1-MB frames to be allocated.

b. IPL the z/TPF system.

3. Enter the ZTEST QUDM command again.

QUDM0014E THE JSON FILE NAME SPECIFIED IS TOO LONG.

Explanation: The file name specified is too long.

System action: The command is rejected.

User response: Specify a file name that is less than 1024 characters in length.

QUDM0015I THE JSON FILE SIZE IS *fsize*, THE TOTAL PAYLOAD SIZE IS *psize*.

Where:
fsize the size of the data in the JSON file
psize the total size of the data sent

Explanation: This is a normal response to the ZTEST QUDM command with the JSON parameter specified.

System action: The z/TPF system sends the JSON data.

User response: None.

QUDM0016E THE JSON PARAMETER WAS SPECIFIED. THE DATATYPE PARAMETER MUST NOT BE QUDM OR QMWR.

Explanation: The JSON parameter was specified on the ZTEST QUDM command along with a data type parameter of QUDM or QMWR. The data type of QUDM is reserved for sawtooth data and the data type of QMWR is reserved for workload simulation data which is similar to the sawtooth data.

System action: The command is rejected.

User response: Specify a data type other than QUDM or QMWR.

QUDM0017E UNABLE TO OPEN THE JSON FILE SPECIFIED. ERROR *errno* REASON *reason*

Where:
errno the errno value returned by the fopen function

reason the error message that is associated with the errno value returned by the fopen function

Explanation: The JSON file could not be opened.

System action: The command is rejected.

User response:
Do the following:

1. Review the errno value for the function to determine the cause of the error.
2. Correct the error.
3. Enter the ZTEST QUDM command again.

QUDM0018E UNABLE TO SEEK THE END OF THE JSON FILE SPECIFIED. ERRNO *errno* REASON *reason*

Where:
errno the errno value returned by the fseek function
reason the error message that is associated with the errno value returned by the fseek function

Explanation: An attempt to find the end of the JSON file was unsuccessful.

System action: The command is rejected.

User response:
Do the following:

1. Review the errno value for the function to determine the cause of the error.
2. Correct the error.
3. Enter the ZTEST QUDM command again.

QUDM0019E UNABLE TO TELL THE ENDING POSITION OF THE JSON FILE SPECIFIED. ERRNO *errno* REASON *reason*

Where:
errno the errno value returned by the ftell function
reason the error message that is associated with the errno value returned by the ftell function

Explanation: An attempt to tell the ending position in the JSON file was unsuccessful.

System action: The command is rejected.

User response:
Do the following:

1. Review the errno value for the function to determine the cause of the error.
2. Correct the error.
3. Enter the ZTEST QUDM command again.

QUDM0020E UNABLE TO OBTAIN MALLOC STORAGE. SIZE=*fsize*

Where:
fsize the size of the 64-bit ECB heap requested

Explanation: An attempt to obtain 64-bit ECB heap for the contents of the JSON file or for the contents of the total payload to be sent was unsuccessful.

System action: The command is rejected.

User response:
Do one of the following:

- Reduce the size of the data being sent:
  - Reduce the amount of data in the JSON file specified.
  - Decrease the number of packages specified.
- Review your 1 MB frame usage.
- Enter the ZCTKA ALTER command with the XMMES parameter specified to change the maximum number of 1 MB frames that an ECB can acquire for 64-bit ECB heap, if necessary.

Enter the ZTEST QUDM command again.

QUDM0021E UNABLE TO READ THE JSON FILE SPECIFIED. ERROR *errno* REASON *reason*

Where:
errno the errno value returned by the fread function
reason the error message that is associated with the errno value returned by the fread function

Explanation: The JSON file could not be read.

System action: The command is rejected.

User response:
Do the following:

1. Review the errno value for the function to determine the cause of the error.
2. Correct the error.
3. Enter the ZTEST QUDM command again.

QUDM0022I THE JSON PAYLOAD WAS SENT SUCCESSFULLY.

Explanation: This is a normal response to the ZTEST QUDM command with the JSON parameter specified.

System action: The z/TPF system sent the JSON data.

User response: None.

QUDM0023E QUDM DRIVER NOT STARTED. ALREADY AT MAXIMUM NUMBER OF INSTANCES.

Explanation: The internal table used to maintain instances of the QUDM driver is full.

System action: The command is rejected.

User response:
Do the following:

1. Stop instances of the driver that no longer need to be run.
2. Start this instance of the driver again.

QUDM0024E QUDM DRIVER NOT STARTED. AN INSTANCE OF THE DRIVER ALREADY EXISTS WITH THE TOKEN SPECIFIED.

Explanation: The QUDM driver was already started with the token specified.

System action: The command is rejected.

User response: Start the QUDM driver again and specify a different token.

QUDM0025I QUDM DRIVER STARTED WITH TOKEN *token*.

Where:
token the token specified on the command

Explanation: This is a normal response to the ZTEST QUDM command with the START parameter specified.

System action: The z/TPF system runs this instance of the QUDM driver in continuous mode.

User response: None.

QUDM0026E STOPPING QUDM DRIVER WITH TOKEN *token*.

Where:
token the token specified when this iteration of the driver was started

Explanation: An error occurred when sending the data.

System action: The z/TPF system ends the QUDM driver.

User response:
Do the following:

1. Review the previous error message.

2. Correct the error.
3. Start this iteration of the driver again.

QUDM0027E A RATE OF ZERO IS NOT VALID.

Explanation: A rate of zero was specified on the ZTEST QUDM command. This is not valid.

System action: The command is rejected.

User response: Start the QUDM driver again and specify a non-zero rate.

QUDM0028E QUDM DRIVER WITH TOKEN *token* IS NOT STARTED.

Where:
token the token specified on the command

Explanation: An attempt was made to stop the iteration of the QUDM driver assigned to the specified token. However, an iteration of the QUDM driver with the token is not started.

System action: The command is rejected.

User response: Correct the token specified on the command.

QUDM0029I QUDM DRIVER WITH TOKEN *token* STOPPED.

Where:
token the token specified on the command

Explanation: This is a normal response to the ZTEST QUDM command with the STOP parameter specified.

System action: The z/TPF system stops running this instance of the QUDM driver.

User response: None.

QUDM0030E NO INSTANCES OF THE QUDM DRIVER ARE STARTED.

Explanation: An attempt was made to stop all instances of the QUDM driver. However, no instances of the QUDM driver are running.

System action: The command is rejected.

User response: None.

QUDM0031E QUDM DRIVER NOT STARTED. THE TOKEN ALL IS NOT ALLOWED.

Explanation: An attempt was made to start an instance of the QUDM driver with the token ALL. However, the token ALL is not allowed because it is a reserved parameter for the ZTEST QUDM command with the STOP parameter specified.

System action: The command is rejected.

User response: Start the QUDM driver again and specify a different token.

QUDM0032I ALL INSTANCES OF THE QUDM DRIVER ARE STOPPED.

Explanation: This is a normal response to the ZTEST QUDM command with the STOP and TOKEN-ALL parameters specified.

System action: The z/TPF system stops running all instances of the QUDM driver.

User response: None.

References

For more information about reading syntax diagrams, also referred to as railroad diagrams, see *Accessibility information* in the TPF Product Information Center.