z/TPF MPIF Driver

User's Guide

This page intentionally left blank.

ZTEST MPIF

MPIF is a TPF feature that allows communication between TPF images across a channel-to-channel (CTOC) device. The data base configuration associated with the TPF images is irrelevant. The CPC's can be sharing a data base (loosely-coupled), or they can have their own data bases. Each TPF CPC is called a MPIF system, while all of the TPF images are referred to as the MPIF complex. The MPIF systems are connected through one or more paths, where each path corresponds to a read/write pair of physical units on each side of the CTOC device.

Each TPF system can have one or more 'users', where a user is a group of system or support programs that send data through MPIF. When 2 users in different CPC's want to communicate they must first establish a connection. This allows a user to know another user is available to send and receive data. Connections are a software mechanism and do not necessarily correspond to a path. There can be more than one user connected over a path, and a user can connect over more than one path (multi-pathing).

TPF also supports Memory-to-Memory (MTOM) paths and connections. This allows users to establish connections in the same MPIF system. MPIF treats these connections much like connections over a CTOC device, and will hide the fact the connection is MTOM from the MPIF applications.

Please see the Multi-Processor Interconnect Facility Reference for more details on MPIF.

The MPIF Test Driver

The MPIF test driver is a tool used to generate communications across a channel-to-channel (CTOC) device. Messages are sent from one MPIF system to another, and the receiving system will acknowledge the messages were received. In addition, both the original messages and the acknowledgment are examined to insure they were not corrupted while being transmitted.

The driver consists of both function test level commands and system test level commands. The function test commands allow testing of individual functions of MPIF such as identifying users, establishing connections, sending data, and forgetting users. The system test level starts MPIF activity with one command, allowing MPIF to be run as background activity.

The MPIF driver is the same type of application a TPF customer might write. The MPIF User Identify Tokens (UITOK) and User Connect Tokens (UCTOK) allow the driver to maintain it's records concerning all users and their connections. The MPIF user exits allow the driver to be aware of the events surrounding it. For example, if one processor in a MPIF complex fails the driver in all other processors will be notified through the MPIF Error Exit (ERX). The driver will then stop sending messages to the failing processor.

The driver can execute on HPO or non-HPO systems, as long the MPIF feature is installed. The only difference when executing on a non-HPO system is the SETUP command is disabled.

The Driver's Structure

The driver maintains 2 different types of records. Each subsystem-user in every MPIF system contains it's own version of these records. The first is the primary record (MPR), which is created when the first user is identified to the system. It's layout is defined by the DCTMPR DSECT. The MPR record consists of slots, one for each user in the system. Each slot contains a pointer to the secondary record for the user.

The other type of record is the driver's secondary records (MUS). There is one of these records for each user in the system. The layout is defined by the DCTMUS DSECT. Each record contains a header which has general information about the user. Following the header there are connection slots, one for each of the user's active connections. Each slot contains information about the connection.

If the user is a MTOM type user both ends of each connection will exist in the same system, and there will be two connection slots for each connection. These slots will be linked together so certain services are not performed twice on the same connection. For example, 'send' processing uses the link to keep from sending message twice over the same connection.

The UITOK a user gives to MPIF is the address of it's secondary block. The UCTOK a user gives to MPIF is the address of the connection slot for the connection. By defining the user tokens in this manner the driver has an easy means of identifying which user and connection are involved when an MPIF exit is kicked off. For example, if a path fails the Error Exit (ERX) will be invoked, and the parameter list will contain the UITOK and UCTOK of the user and connection that has been broken.

Function Test vs. System Test

The test driver has two modes, function test and system test. Function test allows individual functions of MPIF to be tested. One command allows a user to be restarted and to establish connections. Another allows messages to be sent on demand, and the third allows a user to issue a MPIFC FORGET on itself. This means the user will no longer be identified to the system, and all connections with the user are broken. The driver is in function test mode whenever it is not in system test mode (i.e., a ZTEST MPIF START has not been entered). While in function test mode the driver will print all messages concerning it's operation. The driver will say if the user established connections, successfully sent messages, or lost a connection. The operator is kept well informed whenever the driver performs a task, either successfully or unsuccessfully.

The system test level of the driver allows one START command to be issued to begin generating continuous MPIF activity. This is useful for system and regression tests where MPIF needs to be run as background activity. The driver automatically enters system test mode when the START command is issued. While in system test mode most messages concerning the driver's operation are suppressed. The only messages that are printed are error messages. Lost connection messages are also not printed. During system test runs the primary means of monitoring the driver should be the ZTEST MPIF STATUS command.

The system test mode consists of three phases: STARTING, STARTED, and STOPPING. When a ZTEST MPIF START is entered the driver will be in the STARTING phase. It will issue RESTART requests on certain users and then wait for the user's to establish connections. The driver will RESTART 3 MTOM type users, and 3 CTOC type users. Within each user type there is 1 class 'A' user and 2 class 'B' users. After a given amount of time the driver assumes the users have established their connections, and the driver enters the STARTED phase. The driver will then send messages until it is stopped. The driver must be started in at least 2 MPIF systems to generate activity over a CTOC device.

When a ZTEST MPIF STOP is entered the driver enters the STOPPING phase. A global will be set so all ECB's issuing sends will exit, and all of the users that were restarted are forgotten. After a time it is assumed all sending ECB's have exited, and all users have been forgotten. The system test mode is then finished, and the driver returns to function test mode.

The ability of the driver to determine if a user is connected allows the system test mode to be run in any configuration. If system test mode is started on a uni-processor the CTOC type users will be identified, but they won't be able to connect. The driver will see this and not generate any activity on these users. Only MTOM activity will be generated. Similarly, if the complex is loosely-coupled with MPIF IPC, but no class 'B' paths are available, only MTOM and CTOC class 'A' activity will be generated.

The driver has no architectural limits as to the number of processors system test activity can be started in. The driver also does not care if the MPIF systems are sharing a data base. For example, if the complex has three processors the driver can be started in all three, and activity will be generated among all of the processors. It does not matter if the 3 processors have separate data bases, if they are loosely-coupled, or if only two of them are loosely-coupled. Please see the *Complex and System Constraints* section of this document for details on how to set the COMPLEX and SYSTEM parameters to increase the size of the MPIF tables. This may be necessary if the driver is going to be started on more than 3 systems.

The ability of the driver to determine when connections have failed allows it to survive and adapt to events surrounding it. For example, if system test activity is being run in 2 processors and 1 of them fails the other will detect this and stop sending messages to the failing processor. This is done without operator intervention. When the failing processor recovers the driver can be started on it, and cross-system activity will resume on both processors. Another example would be if one processor is put into stop and echo check failures occur on the other processor. Again, the driver will detect this and stop sending messages. When the processor is taken out of stop, and the paths are re-established, the driver will resume sending messages over the CTOC device. NOTE: when the driver does lose connections it is possible some errors will be taken. This is normal because there may be ECB's that have not been made aware of the failure. This is not a problem as long as the cause of the error is known and the errors do not persist for more than a couple of minutes (3 minutes is the minimum time the driver will wait for an acknowledgment).

The driver does allow function test commands to be entered during system test runs. This was done to add flexibility, and to allow the system test run to be tailored to the operator's needs. However, when issuing function test commands during system test runs a great deal of care needs to be taken. Also, since the globals indicate a system test run the driver will suppress most output messages about the function test command. Use the ZTEST MPIF STATUS command to see if the request completed successfully.

Establishing, Breaking, and Re-Establishing Connections

Connections are established through the RESTART function. When a user is restarted they identify themselves to the MPIF system. If the user is a MTOM user it will issue a MPIFC CONNECT to itself. If it is a CTOC user it will query all other systems to see if there are any other users in the complex with the same name. If there is, the users will connect over all available paths. If there is more than one user with the correct name in the complex, connections will be established among all of them. If the driver is in function test mode, messages will be printed to indicate the connections were completed.

If a connection is broken, the driver will automatically update it's records to reflect the change. A connection can be broken due to path failure, a path being stopped, a system IPL'ing itself, or through a FORGET command being issued on one system. In all cases the driver in all systems will be made aware of the change, and they will take action accordingly. No more messages can be sent along a broken connection. If the driver is in function test mode a message will be printed saying the connection has been lost.

If a MPIF path is stopped using the QUIESCE option all user's connected over the path will issue a MPIFC DISCONNECT. This is done because a quiescing path will eventually fail, and any messages on the MPIF queue at the time will be lost. By disconnecting early enough it is hoped none of the driver's messages will be lost.

If the TPF system can recover from the problem that caused the error the connection will be re-established. For example, if a path is stopped the driver will automatically re-establish the connection when the path is re-established. If one system IPL's, or if a user issues a MPIFC FORGET on itself, the connection can be re-established by issuing a RESTART on the system that lost the user. The connection will then be re-established with the system that did not lose the user. MPIF users and their connections do not survive a IPL.

Connections are requested using MPIFC CONNECT, but are not complete until the 'connection complete' exit (QEA6) is kicked off. If a user has requested a connection, but has not received a reply, they are said to have incomplete connections. No messages can be sent by this user until the connection completes. Incomplete connections are indicated by an '*' following the "number of connections" in the DETAIL STATUS display.

Please note that certain WTOPC messages about broken connections and returned MPIF messages are only routed to the prime cras of the 'home' processor. Normally, driver messages are routed to the terminal that issued the last driver command. The exceptions were made because if connections are broken, there is no guarantee the issuing processor is still 'active' (it may be IPL'ing, which caused the break). If the WTOPC is sent to this processor, an OPR-34C will occur.

Sending Messages

When MPIF delivers a message it actually delivers two data areas, data 1 and data 2. Data 1 contains indicators saying such things as how many messages will be sent, and if the message is unsolicited or an acknowledgment. Data 2 contains a unique value in it's first fullword, while subsequent fullwords have a value one greater than the previous fullword. This is done to insure the message is not corrupted during transmission. Whenever any message is received (unsolicited or acknowledgment), data 2 is checked to make sure every full word has a value one greater than the previous fullword. If not, a dump is issued to indicate the data corruption during transmission.

The original 'send' ECB will send the messages, create an event, and wait on it. The messages are acknowledged when the event completes. The DRX will receive the acknowledgment, and issue the POSTC necessary to complete the event.

The user who receives a group of messages will have it's DRX kicked off once for each message sent. However, only one acknowledgment will be sent back, to help reduce channel contention. This is accomplished by having the first ECB to receive a message establish it's own event. It will then wait for all other ECB's who receive the message to post the event. When that is accomplished the acknowledgment is sent to the original sender, who can then post it's own event.

Each group of messages will have a unique event number associated with it. This event number serves three purposes. First, it is the event name used in the EVNTC, EVNWC, and POSTC macros. It is also the value placed in the first fullword of data 2. This allows the DRX to know what event name to post. Finally, the event number is used as tracking mechanism. In function test mode all WTOPC messages concerning MPIF messages and acknowledgments will contain the event number. This allows an association between the output messages and the MPIF messages. The MPIF messages can then be tracked as they travel through the complex.

No messages can be sent by a user with 'incomplete' connections. If a 'SEND' command is entered, it will be rejected with an error message. If system test activity is running, it will bypass the user.

Subsystems and Subsystem-Users

The driver's architecture allows it to run in any subsystem-user, but there are a few restrictions and considerations.

- User names must be unique in the MPIF system. If a user is restarted in one subsystem-user MPIF will not allow it to be restarted in another.
- All functions on a user (such as sending messages) must occur in the same subsystem-user the RESTART was done in.
- ZTEST MPIF STATUS will only show MPIF users restarted in one subsystem-user.
- ZTEST MPIF STATUS ALL will give a summary status report for all subsystem-users. Each subsystem-user has it's own version of the driver's MPR and MUS records.
- System test mode can run in more than one subsystem-user at a time.
- Please see the *Complex and System Constraints* section of this document for details on how to set the COMPLEX and SYSTEM parameters to increase the size of the MPIF tables. This may be necessary if the driver is going to be started in multiple subsystem-users.
- System test activity can run in different subsystem-users on different MPIF systems, but this will only give MTOM activity in each system. The subsystem-users must be the same to generate activity across the CTOC device.

Complex and System Constraints

The ZTEST MPIF SETUP function will issue ZMPIF SET COMPLEX and SYSTEM commands to define such things as maximum number of users and connections. These values are then used by CTIN to carve out MPIF tables. MPIF will not allow these maximums to be exceeded, but the driver will not check if these values will be exceeded when performing a MPIF function. The result is the MPIF tables may not be large enough for the driver to perform a requested function. If the maximum values provided by the SETUP are not adequate then the operator must manually enter ZMPIF SET COMPLEX and SYSTEM commands to increase the maximums. An IPL will be required to allow CTIN to carve out the larger tables. Please see the TPF Operation's Guide for details on these two commands.

Please note the HPO feature must be installed in order for the SETUP function to execute. SETUP uses CPUID's to determine which set of ZMPIF commands to enter. On all 'in-house' base-only systems the CPUID's are 'A', so SETUP has no means of distinguishing them. The operator must manually enter all required ZMPIF commands in lieu of the SETUP function.

Function Test Level Commands

The following commands are the driver's function test level commands. They allow individual functions of MPIF to be tested.

All of the MPIF driver commands start with ZTEST MPIF, which indicates the commands are invoking MPIF driver functions.

All driver commands can be issued in 1052 state or above, except for ZTEST MPIF SETUP, which can be issued before 1052 state.

The following descriptions list the expected response for each command. Please see the *Messages* section of this document for a detailed description of these responses, and for any error responses.

ZTEST MPIF RESTART

Format

```
>>--ZTEST-- --MPIF-- --Restart-- --User-xxxxxxxx--,Class-y--------------->

>--,Device-+-CTOC-+-----------------------------------------------------><
           '-MTOM-'
```

User-xxxxxxxx
is a unique user name of up to 8 characters. The name cannot begin with MPIF or IPC. This parameter must specified since no default is provided.

Class-y
is any valid MPIF class available for use. The available classes can be listed through a ZMPIF DISPLAY CLASS,ALL command. Additional classes can be defined using the ZMPIF SET CLASS command and IPL'ing. The default for this parameter is class 'B', which was chosen since it is a non-IPC path.

Device-(CTOC|MTOM)
is the MPIF device to be used. If DEVICE-CTOC is used connections will be established to other MPIF systems. If DEVICE-MTOM is specified the user will only establish connections in

the same system. In this case the user will connect with itself. The default for this parameter is DEVICE-CTOC.

RESTART

is a driver function that will identify the specified user to the MPIF system. The driver will then query all other systems to determine if a user with the same name has been identified to the MPIF complex. If so, and DEVICE-CTOC was specified, the users will connect. If DEVICE-MTOM was specified the user will only connect with itself.

Expected Response

QEA20003I hh.mm.ss SUCCESSFUL IDENTIFY REQUEST FOR xxxxxxxx

QEA20006I hh.mm.ss SUCCESSFUL CONNECT REQUEST FOR xxxxxxxx TO SYSTEM zzzzzzzz (message expected only if a connection can be established)

QEA50001I hh.mm.ss CONNECTION REQUEST FROM SYSTEM wwwwwwww ACCEPTED BY xxxxxxxx (message expected only if a connection can be established)

QEA60001I hh.mm.ss CONNECTION COMPLETED FOR xxxxxxxx TO SYSTEM zzzzzzzz (message expected only if a connection can be established)

Additional Information

- The user's name must be unique in the MPIF system, including all subsystem-users.
- An error message will be printed if there no paths with the correct class for two users to connect over.
- If there is more than one path available the users will connect over all of the paths.
- If there is more than one user with the correct name in the complex connections will be established to all of them.
- More than one user can connect over the same path.
- The device and class parameters for the user name must consistent throughout the complex.
- The maximum number of users that can be identified in each subsystem-user is 100.
- The maximum number of connections that a user can have is 80. This restriction was imposed to reduce search times during system test runs.
- The IOCP needs to have 2 pairs (CTC or CNC) of addresses defined between each pair of processors that they want to have MPIF connections on

Example

The following example will cause MYUSER to identify itself to the MPIF system. It will be a class 'A' user, and it will only connect with itself since it is a MTOM type user. Since the RESTART is being done in the WP1 subsystem-user, all other ZTEST MPIF commands with this user's name must be done in WP1:

```
WP1/ZTEST MPIF RESTART USER-MYUSER CLASS-A DEVICE-MTOM
```

ZTEST MPIF SEND

Format

```
>>--ZTEST-- --MPIF-- --SENd-- --User-x-- --Size-+-381--+-- ---------->
                                                +-1055-+
                                                '-4K---'
>--Msgs-yyyy-- --Common-+-YES-+-----------------------------------><
                        '-NO--'
```

User-xxxxxxxx
is the user who will send the messages. This parameter must be specified since no default is provided.

Size-(381|1055|4K)
is the block size of the messages to be sent. The default is 1055.

Msgs-yyyy
is the number of messages to be sent. The number must be between 1 and 9999, with a default of 10.

Common-(YES|NO)
specifies whether common or non-shareable blocks should be used to send the messages. The default is NO (use non-shareable blocks).

SEND
is a driver function that will send MPIF messages from a user along all of the it's active connections. The sending user will create an event and wait on it. The event will complete when the acknowledgment is received from the other system. In all cases, messages and acknowledgments are checked to make sure they were not corrupted during transmission. This is done by placing a unique value in the first fullword of the data block. Each subsequent byte will have a value one greater than the value of the previous fullword. When the message is received it is checked to ensure it follows this pattern.

Expected Response

QEA30001I hh.mm.ss ALL MESSAGES SENT SUCCESSFULLY BY xxxxxxxx, NUMBER MESSAGES: yyyy, EVENT: zzzzzzzz, DESTINATION: wwwwwwww

QEAA0002I hh.mm.ss ACKNOWLEDGEMENT SUCCESSFULLY SENT BY xxxxxxxx, EVENT NUMBER zzzzzzzz, DESTINATION SYSTEM vvvvvvvv (this message will appear on the receiving system)

QEA30002I hh.mm.ss ACKNOWLEDGEMENT RECEIVED FOR MESSAGES SENT BY xxxxxxxx, NUMBER MESSAGES: yyyy, EVENT: zzzzzzzz, SOURCE: wwwwwwww

Additional Information

- The value specified in the MSGS parameter indicates the number of messages that will be sent on each of the user's connections. For example, if a user has 4 connections, and a

SEND is issued with MSGS-20, then a total of 80 messages will be sent. If the user is a MTOM type user, messages will only be sent in one direction over each connection. This is despite the fact both ends of the connection exist in the same system, and in the same secondary block.

- This command will be rejected if the user has 'incomplete' connections (a CONNECT request was issued, but the reply has not been received yet).

Example

The following example will cause TESTUSER to send 20 381-byte messages on each of it's connections. Non-shareable blocks will be used:

```
SMPC/ZTEST MPIF SEND USER-TESTUSER,MSGS-20,SIZE-381
```

ZTEST MPIF FORGET

Format

```
>>--ZTEST-- --MPIF-- --Forget-- --User-xxxxxxxx---------------------><
```

User-xxxxxxxx
is the user who will be forgotten. This parameter must be specified since no default is provided.

FORGET
is a driver function that will issue a MPIFC FORGET on the specified user. This will remove the user's name from the MPIF system, and break all of it's connections. The driver will also update it's primary block and release the user's secondary block.

Expected Response

QEA40001I hh.mm.ss FORGET ISSUED FOR xxxxxxxx, RETURN CODE 0000

Additional Information

The FORGET will not be performed if the user is in the process of sending or receiving messages (i.e., waiting for a message acknowledgment, or waiting for all messages to be received so an acknowledgment can be sent). An error message will be printed and the command is rejected. The FORGET can be reissued when the ECB's involved in the sending or receiving of messages have exited.

Example

The following example will cause a MPIFC FORGET to be issued on user USER1. The driver will update it's records to reflect the FORGET:

```
ZTEST MPIF FORGET U-USER1
```

System Test Level Commands

The following commands are the driver's system test level commands. They allow MPIF to be run as background activity during system and regression tests.

All of the MPIF driver commands start with ZTEST MPIF, which indicates the commands are invoking MPIF driver functions.

All driver commands can be issued in 1052 state or above, except for ZTEST MPIF SETUP, which can be issued before 1052 state.

The following descriptions list the expected response for each command. Please see the *Messages* section of this document for a detailed description of these responses, and for any error responses.

ZTEST MPIF START

Format

```
>>--ZTEST-- --MPIF-- --START-- --Level-nnnn,Interval-mm---------------><
```

Level-nnnn
determines the number of MPIF messages to send. The default is 1, which will cause 18 messages to be sent every mm seconds (see the interval parameter). The 18 messages will include 6 381-byte blocks, 6 1055-byte blocks, and 6 4k-byte blocks. Half of the blocks will be common, the rest will be non-shareable. If a higher LEVEL is specified the number of messages will be 18 times 'nnnn'. For example, level=4 will cause 72 messages to be sent, 24 each of 381, 1055, and 4K. The value of 'nnnn' must be between 0 and 9999, with a default of 1.

Interval-mm
determines how often (in seconds) the MPIF messages are sent. The value must be between 1 and 99, with a default of 10.

START
is a driver function that will generate continuous activity. The driver will identify 6 users, and begin to send messages through those users. If a user is not connected, or has 'incomplete' connections, no messages will be sent over it. Every mm seconds one user will send 18*NNNN messages.

Expected Response

QEA10001I hh.mm.ss MPIF DRIVER STARTING, LEVEL=nn, INTERVAL-mm

QEA10002I hh.mm.ss MPIF DRIVER STARTED, LEVEL=nn, INTERVAL-mm

Additional Information

- The driver must be started in each system of the MPIF complex to generate activity over the CTOC device. If a START is issued on one processor then only MTOM activity will be generated on that processor. The other processors will not generate any MPIF activity.
- If the START command is entered while the driver is STARTING or STARTED the only effect will be to change the LEVEL and INTERVAL parameters. The system test level will always have the value of the LEVEL and INTERVAL parameters specified on the latest START. Because of this the parameters can be changed on the fly.
- The LEVEL and INTERVAL parameters are not checked against available system resources. Care must be used when increasing them, since the system can run out of resources, such as core blocks.
- If the driver cannot find a user with completed connections, it will be automatically stopped, since it cannot send any messages.
- The START procedure will identify and try to connect 6 standard system test users, 2 class 'A' and 4 class 'B'. However, the driver will generate activity across any identified and connected user. This gives the ability to generate activity on any path class. For example, if activity is needed on class 'C' use the function test RESTART command to connect a user on class 'C', then issue the START command.
- The driver can be started in more than one subsystem-user.
- The driver must be started in the same subsystem-user in each MPIF system to generate activity over the CTOC device for the subsystem-user.
- Please see the *Complex and System Constraints* section of this document for details on how to set the COMPLEX and SYSTEM parameters to increase the size of the MPIF tables. This may be necessary if the driver is going to be started in multiple subsystem-users, or in more than 3 loosely-coupled processors.
- The driver cannot be started while it is STOPPING. Wait for the driver to complete STOPPING before entering the START command.
- The LEVEL and INTERVAL parameters do not have to be the same in all systems or subsystem-users.
- LEVEL = 0 can be specified on the START command. The driver will perform functions as normal, except it will not send any messages. The operator can use this as a quick way to identify users, or it can be used to 'idle' the driver without the overhead of STOPPING and STARTING again.
- The driver is a lethal utility. This means that TPF cannot cycle while continuous MPIF activity is STARTING, STARTED, or STOPPING.

Example

The following example will start system test activity in the BSS. The parameters imply 90 messages will be sent every 7 seconds, with 30 of each block size:

```
ZTEST MPIF START LEV=5,INTER=7
```

ZTEST MPIF STOP

Format

```
>>--ZTEST-- --MPIF-- --STOP--+-------+---------------------------------><
                             '- -ALL-'
```

ALL
is an optional parameter that will stop the driver in all subsystem-users.

STOP
is the driver function that will stop the system test run. It will set a global indicating ECB's issuing sends should exit. A FORGET is then issued on the system test users, and the system test mode exits.

Expected Response

QEA10006I hh.mm.ss MPIF DRIVER STOPPING

QEA10007I hh.mm.ss MPIF DRIVER STOPPED

Additional Information

- The driver cannot be started during the STOPPING procedure.
- Only the system test user's will be forgotten. Any user's started by function test commands will not be forgotten, but continuous activity across them will be stopped.
- A STOP command can be entered while system test activity is STARTING.

Example

The following example will stop system test activity in the WP2 subsystem-user:

```
WP2/ZTEST MPIF STOP
```

Miscellaneous Commands

The following commands are miscellaneous driver commands.

All of the MPIF driver commands start with ZTEST MPIF, which indicates the commands are invoking MPIF driver functions.

All driver commands can be issued in 1052 state or above, except for ZTEST MPIF SETUP, which can be issued before 1052 state.

The following descriptions list the expected response for each command. Please see the *Messages* section of this document for a detailed description of these responses, and for any error responses.

ZTEST MPIF STATUS

Format

```
>>--ZTEST-- --MPIF-- --STATus--+-------+------------------------------><
                               '- -ALL-'
```

ALL
is an optional parameter to display summary status for all subsystem-users.

STATUS
is a driver command to display the status of the driver. It can be used in function test or system test mode. If ALL is specified, a summary report for all subsystem-users will be printed. Otherwise, a detail report for one subsystem-user will be printed.

Additional Information

- In system test mode all 6 users should be listed. If the complex is defined by the SETUP command the MTOM users should have 2 connections each. If a CTOC device is available the CTOC users should have one connection for each of the other processors the driver is running in.
- In system test mode the MESSAGES SENT and ACKNOWLEDGED columns should steadily increase for each user that has at least one connection. The two columns should be the same as long as no path failures occur.
- An '*' next to the 'number of connections' indicates one or more of the connections is 'incomplete' (the reply to the connect request has not been received yet).

ZTEST MPIF SETUP

Format

```
>>--ZTEST-- --MPIF-- --SETUP-----------------------------------------><
```

SETUP
is a driver command that will automatically define the complex, system, path, device, and class parameters needed to run the driver. The system and MTOM type paths will be defined based on the CPU ID of the processor. CTOC type paths will be defined to all other processors. After the parameters have been set the system will IPL itself. If the parameters are not adequate, the operator must manually enter the needed ZMPIF commands.

Expected Response

QEAE0001W MPIF SETUP BEGINS - RE-IPL WILL OCCUR IN 90 SECONDS

Additional Information

- The SETUP command can only be entered from the BSS subsystem.
- HPO must be installed for SETUP to execute. If the system is base-only there are no unique CPUID's available to distinguish the systems. The required ZMPIF commands must be entered manually.
- The SETUP command can be entered during RESTART (i.e., before the system reaches 1052 state).
- The SETUP command defines parameters for only one processor. The command must be entered on each processor in the complex.

Example

The following example will set-up processor E with CTOC paths to processors B, C, D, Z, 0, 1 and 2:

```
SMPE/ZTEST MPIF SETUP
```

ZTEST MPIF HELP

Format

```
>>--ZTEST-- --MPIF-- --Help--------------------------------------------><
```

HELP
will display a list of driver commands and their parameters.

Source code information

The MPIF driver consists of the following program segments:

Header Files
None.

Macros

Macro	Description
dctmpr.mac	Table that contains the first level record for the driver. It contains the name of each user, and a pointer to it's secondary record (DCTMUS).
dctmus.mac	Table that contains all of the information relevant to one MPIF user.
sd0rv.mac	This macro DSECTs the non-keypointable global record, @ISSDRV, used by I-Stream shared test drivers.

<u>BSOs</u>

Module	Makefile	Segment	Description
QEA0	app_drvs.mak	qea0.asm	Initial segment in the driver. It is invoked whenever a ZTEST MPIF command is entered. It is responsible for parsing and error checking the command, and for calling the appropriate segment to process the request.
QEA1	app_drvs.mak	qea1.asm	Continuous activity segment. It is responsible for issuing the commands to generate continuous MPIF activity.
QEA2	app_drvs.mak	qea2.asm	Handles ZTEST MPIF RESTART commands. It will identify a user to the system, and it will try to connect the user to other users in the complex.
QEA3	app_drvs.mak	qea3.asm	Processes ZTEST MPIF SEND commands. It will create and send messages along all of a user's connections. It will also insure the messages are acknowledged properly.
QEA4	app_drvs.mak	qea4.asm	Handles ZTEST MPIF FORGET requests. It will issue a MPIFC FORGET on a specified user, and update the driver's records accordingly.
QEA5	app_drvs.mak	qea5.asm	Connection Request Exit (CRX). It is invoked for a user when another user in the complex wants to connect. The segment will issue the MPIFC ACCEPT macro needed to accept the request.
QEA6	app_drvs.mak	qea6.asm	Connection Complete Exit (CCX). It is invoked for a user when another user has issued a MPIFC ACCEPT request. The connection is completed and messages can be sent across it.
QEAA	app_drvs.mak	qeaa.asm	Data Received Exit (DRX). It is invoked whenever a user receives a message. The message integrity will be checked to insure the message was not corrupted during transmission, and the acknowledgment to the message will be sent.
QEAB	app_drvs.mak	qeab.asm	Directory Update Exit (DUX). It is invoked for a user when another user with the same name has identified itself to the MPIF complex. This user can now try to establish a connection to the other user.
QEAC	app_drvs.mak	qeac.asm	Path Activation Exit (PAX). It is invoked for a user when a new path is started between two users with the same name. The user's can now connect over the new path.

Module	Makefile	Segment	Description
QEAD	app_drvs.mak	qead.asm	Error Exit (ERX) which. It is called when one of three things happens. If a path is stopped with quiesce the user will issue a MPIFC DISCONNECT to end the connection before the quiesce completes. If MPIF cannot deliver a message it will be returned to the ERX, where an error message will be printed. If a path fails the user will be notified the connection has been lost, and the driver will update it's records accordingly.
QEAE	app_drvs.mak	qeae.asm	Handles ZTEST MPIF SETUP requests. It will issue the ZMPIF commands needed to define the MPIF complex, system, paths, classes, and devices. If these definitions are not adequate the ZMPIF commands will have to be entered manually.
QEAF	app_drvs.mak	qeaf.asm	Handles ZTEST MPIF STATUS requests. If ZTEST MPIF STATUS is entered a detailed report for the subsystem-user will be displayed. If ZTEST MPIF STATUS ALL is entered a summary report for all subsystem-users will be displayed.

CSOs
None.

Messages

In general, driver messages are routed to the terminal that issued the last ZTEST MPIF command for the processor. In addition, some messages are also routed to the PRIME CRAS. These include output messages concerning connections being established and broken, and MPIF messages being sent and acknowledged. Output messages dealing with format errors, such as an invalid parameter, are not routed to the PRIME CRAS unless the last ZTEST MPIF command was issued there.

Certain driver messages, which are noted in the following list, are only routed to the PRIME CRAS of the processor on which the ECB is running. The messages are caused by path failure, which may be caused a processor failure. If the last ZTEST MPIF command was issued on the failing processor, a WTOPC to it would cause OPR-34C dumps.

All of the driver messages are time stamped.

QEA00002E MPIF NOT INSTALLED

Explanation: The MPIF feature is not included in this TPF system.

System Action: None, the ECB immediately exits.

User Response: None, the driver is disabled.

QEA00003E INVALID FOR LOADER GENERAL FILE IPL

Explanation: A driver command was entered after an IPL from a loader general file.

System Action: The command is not executed.

User Response: IPL the prime module, and re-enter the command.

QEA00004E INTERVAL PARAMETER CANNOT BE ZERO

Explanation: A START command was entered with INTERVAL=0.

System Action: The command is not executed.

User Response: Re-enter the command with another INTERVAL parameter.

QEA00005E FORMAT IS ZTEST MPIF STOP (ALL)

Explanation: A STOP message was entered with a parameter other than 'ALL', which is the only valid option.

System Action: The STOP command is not executed.

User Response: Re-enter the command with no options, or with the 'ALL' option.

QEA00006E FORMAT IS ZTEST MPIF STATUS (ALL)

Explanation: A STATUS message was entered with a parameter other than 'ALL', which is the only valid option.

System Action: The STATUS command is not executed.

User Response: Re-enter the command with no options, or with the 'ALL' option.

QEA00007E DEVICE PARAMETER MUST BE CTOC OR MTOM

Explanation: A RESTART command was entered with an invalid device type.

System Action: The RESTART command is not executed.

User Response: Re-enter the command with a correct device type.

QEA00008E MSGS PARAMETER CANNOT BE ZERO

Explanation: A SEND command was entered with MSGS=0.

System Action: The command is not executed.

User Response: Re-enter the command with a correct MSGS parameter.

QEA00009E SIZE PARAMETER MUST BE 381, 1055, OR 4K

Explanation: A SEND command was entered with an invalid message size.

System Action: The SEND command is not executed.

User Response: Re-enter the command with a correct message size.

QEA00010E USER PARAMETER CANNOT BE MPIF OR IPC

Explanation: A function test command was entered with a user name of MPIF or IPC. These are reserved names.

System Action: The command is not executed.

User Response: Re-enter the command with another user name.

QEA00011E FUNCTION CANNOT EXECUTE DURING RESTART

Explanation: A driver command was entered before 1052 state. Only SETUP is allowed to execute during RESTART.

System Action: The command is not executed.

User Response: Re-enter the command when the system reaches 1052 state.

QEA00012E COMMON PARAMETER MUST BE YES OR NO

Explanation: A SEND command was entered with an invalid COMMON parameter.

System Action: The SEND command is not executed.

User Response: Re-enter the command with a correct COMMON parameter.

QEA00013E FUNCTION CAN ONLY EXECUTE ON THE MAIN I-STREAM

Explanation: A command was entered on an application I-stream that is restricted to the main I-stream.

System Action: The command is not executed.

User Response: Re-enter the command on the main I-stream. This is done by entering ZTEST MPIF (options), without specifying an I-stream number, so the default main I-stream is used.

QEA10001I MPIF DRIVER STARTING, LEVEL = xx, INTERVAL = yy

Explanation: The driver start up process has begun due to a START command. The LEVEL and INTERVAL portion of this message is suppressed if the operator used the default values for both.

System Action: The START process begins.

User Response: None.

QEA10002I MPIF DRIVER STARTED, LEVEL = xx, INTERVAL = yy

Explanation: The driver start up process has completed and continuous activity is now STARTED. The LEVEL and INTERVAL portion of this message is suppressed if the operator used the default values for both.

System Action: The driver will continue to send MPIF messages.

User Response: None.

QEA10003E MPIF DRIVER IS STOPPING, START REJECTED

Explanation: A START command was entered while the driver was STOPPING.

System Action: The STOPPING process continues.

User Response: Wait for the driver to complete STOPPING and re-enter the START command.

QEA10004E NO CONNECTIONS FOR MPIF DRIVER TO EXECUTE

Explanation: The driver has no connections to generate continuous activity over.

System Action: The driver is automatically stopped.

User Response: Determine why the connections were lost and try to re-establish them before restarting the driver.

QEA10005E MPIF DRIVER IS ALREADY STOPPING OR STOPPED

Explanation: A STOP message was entered while the driver was already STOPPING or STOPPED.

System Action: None, the command is ignored.

User Response: None.

QEA10006I MPIF DRIVER STOPPING

Explanation: A STOP command was entered, and the driver is now in the STOPPING phase.

System Action: The driver sets a global to indicate all SENDs are to stop.

User Response: None.

QEA10007I MPIF DRIVER STOPPED

Explanation: The driver STOPPING phase has completed. System test mode is now STOPPED.

System Action: None, continuous activity has stopped.

User Response: None.

QEA20001E CANNOT RESTART xxxxxxxx - MAXIMUM NUMBER OF USERS EXCEEDED

Explanation: A RESTART command was entered which will cause the maximum number of users in the subsystem-user to be exceeded.

System Action: The RESTART command is not executed.

User Response: Issue a FORGET on another user to allow user xxxxxxxx to be RESTARTED.

QEA20002E xxxxxxxx HAS ALREADY BEEN RESTARTED

Explanation: A RESTART was issued but user xxxxxxxx has already been identified to the system. This message is suppressed during system test runs.

System Action: The RESTART command is not executed.

User Response: Re-enter the command with another user name.

QEA20003I SUCCESSFUL IDENTIFY REQUEST FOR xxxxxxxx

Explanation: User xxxxxxxx has been successfully identified to the MPIF system.

System Action: This message is suppressed during system test runs. The driver creates a secondary block for the user and updates it's records to reflect the new user.

User Response: None.

QEA20004E NO CLASS y PATHS TO zzzzzzzz, NO CONNECTION MADE FOR xxxxxxxx

Explanation: User xxxxxxxx wants to connect to user xxxxxxxx in system zzzzzzzz, but there are no class y paths available to make the connection.

System Action: No connections are established, but the user is still identified to the system.

User Response: Start a class y path to system zzzzzzzz so the connection can be made.

QEA20005E CONNECT REQUEST SUPPRESSED FOR xxxxxxxx TO SYSTEM zzzzzzzz

Explanation: User xxxxxxxx wants to issue a MPIFC CONNECT to system zzzzzzzz, but doing so would cause the user to exceed it's maximum number of connections.

System Action: No MPIFC CONNECT is issued.

User Response: If this connection is needed, then halt another connection so this one can be established.

QEA20006I SUCCESSFUL CONNECT REQUEST FOR xxxxxxxx TO SYSTEM zzzzzzzz

Explanation: User xxxxxxxx has issued a connection request to xxxxxxxx in system zzzzzzzz. This does not mean the connection has been established. This message is suppressed during system test runs.

System Action: None, the Connection Complete Exit (CCX) must be kicked off to indicate the other user has accepted the request.

User Response: None.

QEA20007E MPIFC IDENTIFY RETURN CODE yy FOR xxxxxxxx

Explanation: User xxxxxxxx issued a MPIFC IDENTIFY but a non-zero condition code was returned. The return code is in decimal.

System Action: Unknown, the user may or may not be identified. No secondary block is created for the user. It is recommended the user's name not be used until after the next IPL.

User Response: Look up the condition codes under the MPIFC macro in the TPF System Macros Manual to determine what caused the error.

QEA20008E MPIFC QUERY RETURN CODE yy FOR xxxxxxxx

Explanation: User xxxxxxxx issued a MPIFC QUERY but a non-zero condition code was returned. The return code is in decimal.

System Action: Unknown, the driver may or may not have been able to establish some of the connections for the user.

User Response: Look up the condition codes under the MPIFC macro in the TPF System Macros Manual to determine what caused the error.

QEA20009E MPIFC CONNECT RETURN CODE yy FOR xxxxxxxx

Explanation: User xxxxxxxx issued a MPIFC CONNECT but a non-zero condition code was returned.

System Action: Unknown, the driver may or may not have been able to establish the connection for the user. The return code is in decimal.

User Response: Look up the condition codes under the MPIFC macro in the TPF System Macros Manual to determine what caused the error.

QEA30001I ALL MESSAGES SENT SUCCESSFULLY BY xxxxxxxx, NUMBER MESSAGES: yyyy, EVENT: zzzzzzzz, DESTINATION: wwwwwwww

Explanation: User xxxxxxxx sent yyyy messages to system wwwwwwww. The event number uniquely identifies the group of messages, and can be used to track the messages as they travel through the complex. This message is suppressed during system test runs.

System Action: The secondary block for the user is updated to indicate yyyy messages were sent successfully.

User Response: None.

QEA30002I ACKNOWLEDGEMENT RECEIVED FOR MESSAGES SENT BY xxxxxxxx, NUMBER MESSAGES: yyyy, EVENT: zzzzzzzz, SOURCE: wwwwwwww

Explanation: User xxxxxxxx received the acknowledgment for the messages identified by event zzzzzzzz. The acknowledgment was received from system wwwwwwww. This message is suppressed during system test runs.

System Action: The secondary block for the user is updated to indicate yyyy messages were acknowledged.

User Response: None.

QEA30003E xxxxxxxx DOES NOT EXIST, NO MESSAGES SENT

Explanation: A SEND was issued but the specified user was not RESTART'ed in this subsystem-user.

System Action: No messages are sent.

User Response: Re-issue the command with another user's name.

QEA30004E xxxxxxxx IS NOT CONNECTED TO ANOTHER USER

Explanation: A SEND was issued but the specified user is not connected to another user.

System Action: No messages are sent.

User Response: Reissue the command with another user's name, or re-establish a connection for user xxxxxxxx.

QEA30005E MPIFC SEND RETURN CODE yy FOR xxxxxxxx, EVENT NUMBER zzzzzzzz, DESTINATION SYSTEM aaaaaaaa

Explanation: User xxxxxxxx issued a MPIFC SEND to system aaaaaaaa, but a non-zero condition code was returned. Event number zzzzzzzz identifies the group of messages being sent when the failure occured.

System Action: Unknown. The driver may have sent some or all of messages specified in the SEND command. The driver will update the user's secondary block to reflect any messages that were sent, but it will not wait for an acknowledgment. The return code is in decimal. This message is suppressed if the return code is 8 during a system test run.

User Response: Look up the condition codes under the MPIFC macro in the TPF System Macros Manual to determine what caused the error. Additional errors should be expected, especially QEAA0003E on the receiving system.

QEA30006E MESSAGES SENT BY xxxxxxxx WERE NOT ACKNOWLEDGED, NUMBER MESSAGES: yyyy, EVENT: zzzzzzzz, SOURCE: wwwwwwww

Explanation: User xxxxxxxx did not receive the acknowledgment it requested for the group of messages identified by event zzzzzzzz. This message is suppressed during system test runs.

System Action: None.

User Response: Look at the consoles for the sending and receiving systems. Trace the course of the messages and determine at what point the acknowledgment mechanism failed.

QEA30007E xxxxxxxx HAS INCOMPLETE CONNECTIONS

Explanation: User xxxxxxxx has one or more 'incomplete' connections (the reply to the CONNECT request has not been received yet). Messages cannot be sent over such a user.

System Action: No messages are sent.

User Response: Wait for the connections to complete, and reissue the SEND command.

QEA40001I FORGET ISSUED FOR xxxxxxxx, RETURN CODE yy

Explanation: User xxxxxxxx issued a MPIFC FORGET on itself, and the return code was yy. A return code of zero indicates the FORGET was successful. This message is suppressed during system test runs if the return code is zero. The return code is in decimal.

System Action: The user's secondary block is released and the driver's primary block is updated to reflect the FORGET.

User Response: If the return code is not zero then look up the condition codes under the MPIFC macro in the TPF System Macros Manual to determine what caused the error. No action is required if the return code is zero.

QEA40002E xxxxxxxx DOES NOT EXIST, FORGET NOT ISSUED

Explanation: A FORGET was issued but the specified user was not RESTARTED in this subsystem-user.

System Action: None, the command is rejected.

User Response: Reissue the command with another user's name.

QEA40003E xxxxxxxx HAS ACTIVE ECB's IN THE SYSTEM, FORGET NOT ISSUED

Explanation: User xxxxxxxx has active ECB's in the system which are involved in the SEND and acknowledgment process. The FORGET cannot be issued until these ECB's exit.

System Action: None, the command is rejected.

User Response: Reissue the command when the ECB's complete.

QEA50001I CONNECTION REQUEST FROM SYSTEM wwwwwwww ACCEPTED BY xxxxxxxx

Explanation: User xxxxxxxx has received and accepted a request to connect to system wwwwwwww. This message is suppressed during system test runs.

System Action: The driver assumes the connection is completed and updates the user's secondary block to reflect the new connection.

User Response: None.

QEA50002W CONNECTION REQUEST FROM SYSTEM wwwwwwww REJECTED BY xxxxxxxx

Explanation: User xxxxxxxx received a connection request from system wwwwwwww they had to reject. The reason is if the connection was accepted the user would exceed it's maximum number of connections.

System Action: A MPIFC DISCONNECT is issued to reject the request.

User Response: If this connection is needed then halt another connection so this one can be established.

QEA50003E MPIFC ACCEPT RETURN CODE yy FOR xxxxxxxx

Explanation: User xxxxxxxx issued a MPIFC ACCEPT but a non-zero condition code was returned. The return code is in decimal.

System Action: Unknown, the connection request may or may not have been accepted.

User Response: Look up the condition codes under the MPIFC macro in the TPF System Macros Manual to determine what caused the error.

QEA50004E MPIFC DISCONNECT RETURN CODE yy FOR xxxxxxxx

Explanation: User xxxxxxxx issued a MPIFC DISCONNECT but a non-zero condition code was returned. The return code is in decimal. The DISCONNECT was issued to reject a connection request.

System Action: Unknown, the driver may or may not have been able to reject the request.

User Response: Look up the condition codes under the MPIFC macro in the TPF System Macros Manual to determine what caused the error.

QEA60001I CONNECTION COMPLETED FOR xxxxxxxx TO SYSTEM wwwwwwww

Explanation: A connection requested by user xxxxxxxx made to system wwwwwwww has been accepted successfully. This message is suppressed during system test runs.

System Action: The user's secondary block is updated to reflect the completed connection.

User Response: None.

QEA60002E CONNECTION NOT ESTABLISHED FOR xxxxxxxx, COMPLETION VALUE yyyy

Explanation: A connection request for user xxxxxxxx did not complete successfully. The reason is contained in the completion value. The completion value is in hexadecimal.

System Action: None.

User Response: Look up the completion value in the DCTMUP macro to determine the cause of the error. The completion values are in the CCECV field of the Connection Complete Exit of DCTMUP.

QEA60003E CONNECTION NOT ESTABLISHED FOR xxxxxxxx, CONNECTION REQUEST WAS REJECTED

Explanation: A connection request user xxxxxxxx initiated was rejected using a MPIFC DISCONNECT.

System Action: The user's secondary block is updated to reflect the connection has not been established.

User Response: The reason for the rejection is not indicated on this system. The reason must be found on the system that issued the rejection, and appropriate action should be taken there. The most likely reason is the other user would have exceeded it's maximum number of connections.

QEAA0001E xxxxxxxx RECEIVED AN UNSOLICITED ACKNOWLEDGEMENT, EVENT NUMBER yyyyyyyy, SOURCE SYSTEM wwwwwwww

Explanation: User xxxxxxxx received an acknowledgment for a group of messages identified by event yyyyyyyy. The problem is the user is not waiting for such an acknowledgment. This message is suppressed during system test runs.

System Action: None.

User Response: Look at the consoles for the sending and receiving systems. Trace the course of the messages and determine at what point the acknowledgment mechanism failed. The most likely cause for this error is the user did send these messages, but the event that was established to wait for the acknowledgment timed out.

QEAA0002I ACKNOWLEDGEMENT SUCCESSFULLY SENT BY xxxxxxxx, EVENT NUMBER yyyyyyyy, DESTINATION SYSTEM wwwwwwww

Explanation: User xxxxxxxx has received all of the messages in the yyyyyyyy group and has successfully sent the acknowledgment. This message is suppressed during system test runs.

System Action: An acknowledgment message has been sent to system wwwwwwww for the yyyyyyyy messages.

User Response: None.

QEAA0003E xxxxxxxx UNABLE TO SEND MESSAGE ACKNOWLEDGEMENT, EVENT NUMBER yyyyyyyy, DESTINATION SYSTEM wwwwwwww

Explanation: User xxxxxxxx received one or more messages in the group of messages identified by event yyyyyyyy, but the event that was established to wait for all of the messages has timed out. This message is suppressed during system test runs.

System Action: None, no acknowledgment is sent.

User Response: Look at the consoles for the sending and receiving systems. Trace the course of the messages and determine at what point the acknowledgment mechanism failed. The most likely cause is the sending system took an error that did not allow it to send all of the messages.

QEAA0004E

Same as QEA30005E

QEAB0001E

Same as QEA20005E

QEAB0002I

Same as QEA20006I

QEAB0003E

Same as QEA20008E

QEAB0004E

Same as QEA20004E

QEAB0005E

Same as QEA20009E

QEAC0001E

Same as QEA20005E

QEAC0002I

Same as QEA20006I

QEAC0003E

Same as QEA20008E

QEAC0004E

Same as QEA20009E

QEAD0001E MPIF UNABLE TO SEND MESSAGE FOR USER xxxxxxxx

Explanation: A message xxxxxxxx had sent successfully has been returned because MPIF could not transmit it. The most likely reason is a MPIF path failed while the message was on the queue waiting to be transmitted. This output message is only routed to the PRIME CRAS of the processor executing the ECB.

System Action: The message blocks are released.

User Response: Determine the cause if it is not immediately apparent (e.g., path failure) and correct the error.

QEAD0002W SUCCESSFUL DISCONNECT FOR xxxxxxxx TO SYSTEM wwwwwwww

Explanation: The user has issued a MPIFC DISCONNECT for one of it's connections to system wwwwwwww. The reason is the path to that system has started to quiesce. The DISCONNECT is issued so the quiesce can complete cleanly. This message is suppressed during system test runs.

System Action: The driver updates the user's secondary record to reflect the lost connection.

User Response: None.

QEAD0003E MPIFC DISCONNECT RETURN CODE yy FOR xxxxxxxx

Explanation: User xxxxxxxx issued a MPIFC DISCONNECT but a non-zero condition code was returned. The return code is in decimal.

System Action: Unknown, the driver may or may not have been able to disconnect the connection.

User Response: Look up the condition codes under the MPIFC macro in the TPF System Macros Manual to determine what caused the error.

QEAD0004W CONNECTION LOST TO SYSTEM wwwwwwww FOR USER xxxxxxxx

Explanation: A connection for user xxxxxxxx to system wwwwwwww has been broken. The most likely reason is the path which had the connection has failed. This message is suppressed during system test runs, and is only routed to the PRIME CRAS of the processor executing the ECB.

System Action: The driver will update the user's secondary block to reflect the lost connection.

User Response: If the path was intentionally stopped this message can be ignored. If the path failed without operator intervention then determine the cause of the failure and try to restart the path so the connection can be re-established.

QEAD0005W xxxxxxxx IS NO LONGER CONNECTED TO ANOTHER USER

Explanation: This message will follow message QEAD0004W when the lost connection was the only connection the user had left. This message is suppressed during system test runs, and is only routed to the PRIME CRAS of the processor executing the ECB.

System Action: The driver will not allow the user to send any messages until it has re-established at least one connection.

User Response: None.

QEAD0006W DISCONNECT FOR xxxxxxxx TO SYSTEM yyyyyyyy SUPPRESSED

Explanation: A warning was received indicating the path to system yyyyyyyy is quiescing. User xxxxxxxx has a connection over the path, but the connection is 'incomplete' (the reply has not been received yet). Since the state of the connection is questionable, the normal DISCONNECT processing done at this time has been bypassed.

System Action: None. If the connection completes, it will be broken again when the path finishes quiescing. If the connection does not complete, the CONNECT reply will be returned in error when the path stops. This will notify the driver that the connection failed.

User Response: None.

QEAE0001W MPIF SETUP BEGINS - RE-IPL WILL OCCUR IN 90 SECONDS.

Explanation: A ZTEST MPIF SETUP has been entered to initialize the MPIF complex, system, device, class, and path parameters.

System Action: The driver will issue the ZMPIF commands needed to define the MPIF parameters. It will then force an IPL to pick up the new parameters.

User Response: Wait for the IPL before doing any further work.

QEAE0002E SETUP IS NOT ALLOWED ON A BASE-ONLY SYSTEM

Explanation: ZTEST MPIF SETUP was entered on a system which is non-SDPS (unable to loosely-couple). The SETUP function is disabled in this environment since there are no CPUID's to uniquely identify the systems.

System Action: None, the SETUP is rejected.

User Response: The ZMPIF commands must be entered manually. Please see TPF Operations for information on the required parameters. The commands listed in QEAE can be used as a guide, but caution must be used.

QEAE0003E SETUP IS RESTRICTED TO THE BSS

Explanation: A ZTEST MPIF SETUP was entered in a subsystem other than the BSS. The SETUP must be done from the BSS because ZMPIF commands are restricted to the BSS.

System Action: None, the SETUP is rejected.

User Response: Reissue the SETUP in the BSS.

System Errors

EEA001

Explanation: No users found on a MPIFC QUERY. The QUERY should have at least found the user that issued the request.

Issued By: QEA2

User Action: Find the reason why the query failed. No connections will be established during the RESTART process.

Message Appended: NO USERS FOUND ON A MPIFC QUERY REQUEST

EEA002

Explanation: An EVNTC macro was issued using an event name already in use.

Issued By: QEA3

User Action: Find the reason why the error occurred. A user cannot send messages while the error persists.

Message Appended: EVENT NAME USED BY EVNTC MACRO ALREADY IN USE

EEA003

Explanation: An EVNWC or POSTC macro was issued but the event name was not found.

Issued By: QEA3, QEAA

User Action: Find the reason why the error occurred. The driver's message acknowledgment mechanism cannot function if this error persists.

Message Appended: EVENT NAME USED BY EVNWC OR POSTC NOT FOUND

EEA004

Explanation: A message delivered to a MPIF user was corrupted during transmission.

Issued By: QEAA

User Action: Examine the dump and determine how the message was corrupted. Investigate the reason and correct the problem. The driver's message acknowledgment mechanism may fail due to this error.

Message Appended: CORRUPTED MESSAGE RECEIVED BY THE DRX

EEA005

Explanation: The system clock could not be stored into memory.

Issued By: QEAB, QEAC

User Action: Examine the dump and determine what condition code was returned by the STCK instruction. MPIF user's may not connect properly if this error is issued.

Message Appended: ERROR STORING TOD CLOCK

EEA006

Explanation: QEA1 was entered with an invalid value in EBW000. The value in EBW000 normally indicates the service QEA1 should perform.

Issued By: QEA1

User Action: Examine the dump and determine what value was contained in EBW000 and who placed it there. QEA1 will not perform any services if this error is issued.

Message Appended: INVALID ENTRY TO THE CONTINUOUS ACTIVITY SEGMENT QEA1

EEA007

Explanation: A ZTEST MPIF STATUS command was entered but QEAF could not decode the system test activity indicators to see what state the driver is in.

Issued By: QEAF

User Action: Examine the dump and determine what values are in the system test indicators and who placed them there. The STATUS request will not be completed.

Message Appended: CAN NOT DECODE CONTINUOUS ACTIVITY INDICATORS FOR A STATUS REQUEST

References

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