

z/OS Message Flood Automation User's Guide

Table of Contents

- **z/OS Message Flood Automation User's Guide**
 - Introduction
 - Operation
 - Message Flood Automation and CONSOLxx parameters
 - Message Flood Automation and WTO processing
 - Message Flood Automation and MPFLSTxx parameters
 - Message Flood Automation and the Subsystem Interface (SSI)
 - Message Flood Automation and the Console Restructure
 - Message Flood Automation and EMCS consoles
 - Limitations
 - Operator Commands
 - Re-initialize the Message Flood Automation parameters
 - Enable message flood checking
 - Disable message flood checking
 - Display individual Message Flood Automation parameters
 - Display all of the Message Flood Automation parameters
 - Display all of the Message Flood Automation default actions
 - Display the default job actions for REGULAR and ACTION messages
 - Display the default actions for SPECIFIC messages
 - Display the Message Flood Automation status
 - Display the Message Flood Automation intensive mode states
 - Modify individual Message Flood Automation parameters
 - Modify the Message Flood Automation default actions
 - Modify the job actions for REGULAR and ACTION messages
 - Modify the actions for SPECIFIC messages
 - Release the Message Flood Automation common storage (SQA) area
 - Enable message rate monitoring
 - Disable message rate monitoring
 - Display message rate information
 - Setting thresholds based on message rates
 - Message flood detection behavior
 - Command Summary
 - DISPLAY MSGFLD command
 - SET MSGFLD command
 - SETMF command
 - Built-in defaults
 - PARMLIB specifications
 - comment statements
 - msgtype statements
 - DEFAULT statements
 - DEFAULTCMD statements
 - JOB statements
 - MSG statements

- Exempting messages from processing by Message Flood Automation
- Sample PARMLIB specification
- ABEND codes
 - ABEND X'077' RSN X'039'
 - ABEND X'077' RSN X'A24'
- Operator Messages
- SYSLOG records
- SYSLOG MPF Flags
- SYSLOG Message ordering
- Recovery
- Other Information
- Terms and Conditions
- **Installation**
 - Instructions for installing Message Flood Automation
 - Criteria for selecting either CNZZVXT1 or CNZZVXT2
 - Assembling CNZZVXT1 or CNZZVXT2
 - Linking CNZZVXT1 or CNZZVXT2 with CNZZVMXT
 - Creating an SMP/E ++USERMOD
 - Using CNZZCMXT
 - Using IEAVMXIT
 - Providing a MSGFLDxx member of PARMLIB
 - Considerations when migrating from one level to another
 - MSGFLDxx PARMLIB considerations
 - Instructions for initializing Message Flood Automation.
 - Instructions for shutting down Message Flood Automation.

Appendix

- Operator command changes from Message Flood Automation 1.2.x
- Operator command changes from Message Flood Automation 1.3.x/1.4.x
- Customization

z/OS Message Flood Automation User's Guide

11 July 2007 V2R0M00

Licensed Materials - Property of IBM

(C) Copyright IBM Corporation 1990,2007 - All Rights Reserved.

US Government Users Restricted Rights - Use, duplication or disclosure restricted by GSA ADP Schedule Contract with IBM Corp.

Introduction

Many z/OS systems are troubled by cases where a user program or a z/OS process itself issues a large number of messages to the z/OS consoles in a short time. For example, a user program may enter an unintentional loop that includes a WTO call, with the result that a potentially infinite number of messages are issued in a short time. Cases of hundreds (or even thousands) of messages a second are not uncommon.

These messages are often very similar or identical, but are not necessarily so. Techniques to identify similar messages can be very difficult and time consuming.

Message Flood Automation addresses this problem. Note that it does not claim to identify all cases of erroneous behaviour, nor to take the 'correct' action in all cases. Its intention is to identify runaway WTO conditions that can cause severe disruptions to z/OS operation and to take installation-specified actions in these cases.

Disruptions are caused because:

- Large numbers of messages to the z/OS consoles can obscure important messages and delay them from being acted on.
- Large numbers of messages to the automation system (e.g. NetView) can delay the processing of normal messages.
- Messages can use excessive CPU and storage resources. Buffering excessive message traffic may use large amounts of virtual and real storage and it can cause SQA to overflow into CSA. This can cause jobs, subsystems and even complete systems to be delayed or even to fail.

Message Flood Automation can react to potential message flooding situations in a matter of tens or hundreds of messages (specifiable by the installation), well before buffers have begun to fill, well before console queues have begun to build, and well before console message rates have begun to skyrocket. Furthermore, its actions do not result in residual buffers or queues of messages that must be "worked down" to return to normal processing. Because its processing is targeted to the messages that are causing the problem, very few uninvolved messages are acted upon. By contrast, the act of flushing console queues (with the K Q command) can result in throwing away many innocent and often important messages.

Message Flood Automation is unable to handle DOM (Delete Operator Message) floods.

Operation

Message Flood Automation is implemented as a Message Processing Facility (MPF) IEAVMXIT routine that is called as a part of z/OS WTO processing. The exit examines each message in the z/OS system, and attempts to identify when too many WTOs are being issued and by whom. It then takes appropriate action - usually to suppress the message from display at a z/OS console, and to indicate that automation processing is not required. It can also issue commands e.g. to cancel the user or process.

Three separate classes of messages are handled. These classes are:

- SPECIFIC messages: that is, a set of messages identified by the installation that are to be handled separately.
- ACTION messages: that is, messages that have one or more of the following descriptor codes set:
 - 1 - system failure messages (typically messages with a W message ID suffix)
 - 2 - immediate action required messages (typically messages with an A or D message ID suffix)
 - 3 - eventual action required messages (typically messages with an E message ID suffix)
 - 11 - critical eventual action required messages (typically messages with an E message ID suffix)
- REGULAR messages: that is, messages that do not fall into any of the above categories.
NOTE: REGULAR messages include the "command echo" messages which put the text of a command into the display area of a console and place the text of a command into the SYSLOG and OPERLOG.

Each class of messages is handled separately. Each class has its own set of controls (MSGTHRESH, INTVLTIME, etc.) Each set of controls operate independently; for example, the system may be in intensive mode for regular messages but not for action messages. The effect is that action messages will still be processed by z/OS in the normal way.

Message Flood Automation can take action against "privileged" messages which are queued to consoles even in storage shortage situations.

Message Flood Automation runs in two modes: *normal* and *intensive*.

- In *normal* mode, messages are counted. When a threshold number (MSGTHRESH) of messages have been counted, the *time* taken to count those messages is determined. If the time is less than a limit value (INTVLTIME), the system is placed into *intensive* mode. It is expected that this determination is likely to be done relatively infrequently e.g. every 50-100 messages or more. The INTVLTIME value should be set to identify high message rates e.g. a value of 5 seconds for INTVLTIME indicates an average rate of 20 messages per second if MSGTHRESH is set to 100.

The processing overhead in normal mode is therefore very small. Only a very small number of instructions are executed in the exit for each message. No dynamic storage is obtained or freed and no recovery environment is established.

- In *intensive* mode, each message is subject to extra processing. Messages are counted for each address space (up to a maximum of 10) issuing messages and compared to a further limit value (JOBTHRESH). If any one address space issues JOBTHRESH messages within INTVLTIME then it is subject to action from that time on. This action may be installation-specified, but is typically defaulted to be no-display and no-automation.

At the end of each interval of MSGTHRESH messages a check is made to see if intensive mode should be maintained, and whether address space(s) in 'act-upon mode' should remain so.

Message bursts can end suddenly. The address space that issues them may suddenly exit a tight-loop condition and resume normal processing. In this circumstance, it is likely that subsequent messages are important and should be processed normally. To allow this to happen, there are two further controls: system inter-message time (SYSIMTIME) and job (or message) inter-message time (JOBIMTIME or MSGIMTIME).

When in intensive mode, if the time since the last message is greater than SYSIMTIME, then intensive mode is discontinued. This ensures that the first message after a break is not acted upon.

Similarly, if an address space is in act-upon mode, and the time since its last message exceeds the JOBIMTIME, then it is removed from act-upon mode.

For specific messages, if a message is in act-upon mode, and the time since the last message exceeds the MSGIMTIME, then the message is removed from act-upon mode.

The control algorithms for regular and action messages are identical and are as described above. For specific messages, the control algorithm is similar although it is applied to individual messages and not to jobs or address spaces. The MSGLIMIT parameter performs the same function in specific message processing that the JOBTHRESH parameter performs in regular and action message processing. The MSGIMTIME parameter performs the same function in specific message processing that the JOBIMTIME parameter performs in regular and action message processing although it is applied against specific messages rather than address spaces.

Message Flood Automation and CONSOLxx parameters

CONSOLxx members of PARMLIB may contain statements that can affect the routing and hardcopying of messages.

During WTO processing, the route codes defined by the DEFAULT statement are applied to *unsolicited* messages which have been issued:

- without route codes
- without descriptor codes
- without a console ID or console name

If the ROUTCODE parameter is not supplied on the DEFAULT statement, route codes 1-16 are applied.

Message Flood Automation can affect messages given route codes by DEFAULT processing just like messages that have been issued with route codes, descriptor codes or console routing information.

The HARDCOPY statement defines the route codes of the messages that are subject to being hardcopied. Messages with route codes 1, 2, 3, 4, 7, 8, 10 and 42 are always hardcopied whether the ROUTCODE parameter is supplied on the HARDCOPY statement or not.

Message Flood Automation can affect the hardcopying of messages that have been forced to hardcopy by HARDCOPY processing just like messages that were explicitly issued to hardcopy.

Message Flood Automation and WTO processing

Message Flood Automation runs as part of MPF processing, which occurs after the control block that represents the message has been created. Message Flood Automation is able to see and alter any processing of the message that occurs prior to the creation of this control block. For example, some automation products replace the Write-To-Operator (WTO) Supervisor Call (SVC) with their own code and then invoke the WTO code when they are finished. Message Flood Automation is able to see and react to messages that have been "front-ended" by other automation in this way.

Message Flood Automation and MPFLSTxx parameters

MPFLSTxx members of PARMLIB may contain statements that can affect the display, automation and retention of messages.

During MPF processing, the RETAIN, AUTO and SUP parameters on an MPFLSTxx entry are processed first. Then *either* a user exit (specified by the USEREXIT parameter) is invoked *or* IEAVMXIT is invoked -- *but not both*.

Note that if an MPFLSTxx entry does not exist for a message, the settings from the NO_ENTRY specification are applied.

Since Message Flood Automation message processing runs as IEAVMXIT, it has the ability to override the RETAIN, AUTO and SUP specifications set by the MPFLSTxx entry for the message or set by the NO_ENTRY specification.

Message Flood Automation cannot override specifications set by individual MPF exits.

Message Flood Automation and the Subsystem Interface (SSI)

Message Flood Automation processing occurs before a message is placed onto the Subsystem Interface (SSI). Automation products such as NetView, which can obtain messages from the subsystem interface, will see messages after they have been seen (and potentially modified) by Message Flood Automation. Since all requests to delete, log or queue messages are processed *after* return from the subsystem interface, NetView and other automation products which sit on the subsystem interface are able to see the message and potentially copy or modify the message (possibly overriding Message Flood Automation decisions) *before* z/OS deletes, logs or queues the message.

The NetView 5.2 Message Revision Table (MRT) performs all of its message processing on the Subsystem Interface, after the message has been processed by Message Flood Automation. Message Revision Table logic can see message specification changes requested by Message Flood Automation and can override them. Changes to the message's specifications requested by the Message Revision Table can affect the logging, display, retention and automation of the message by z/OS.

NetView can obtain messages for automation through either the Subsystem Interface or through an EMCS console interface or both. (If the NetView MSGIFAC parameter is set to SSIEXT, USESSI, QUESSI or QSSIAT, NetView will obtain messages for processing from the Subsystem Interface. If the MSGIFAC parameter is set to SSIEXT, only unsolicited messages are obtained from the SSI; command

response messages are obtained through the EMCS console interfaces). When NetView obtains messages from the Subsystem Interface, it obtains a *copy* of the original message, before z/OS has had an opportunity to delete, log or queue the message to a console. The original message is processed for deletion, logging and queuing *after* NetView has made its copy. Traditional (non-Message Revision Table) NetView automation can see but cannot alter changes to the message made by Message Flood Automation.

Message Flood Automation and the Console Restructure

The Console Restructure moves the point from which messages are logged from after they had been displayed on all of the consoles to immediately after the message returns from the Subsystem Interface (SSI). Message Flood Automation occurs before the logging decision is made and can prevent the message from being written to the log (either SYSLOG or OPERLOG).

The Console Restructure moves the point from which messages are sent to other members of the sysplex from after they had been displayed on all of the consoles to immediately after the message returns from the Subsystem Interface (SSI). Message Flood Automation occurs before the decision to send the message to other members of the sysplex is made and can prevent the message from being propagated to other members of the sysplex. The Console Restructure uses XCF to send messages not only to other members of the sysplex but also to send the messages from the message issuer's address space to the Consoles address space. Message Flood Automation occurs before this happens and can prevent messages from ever being sent to the Consoles address space.

The Console Restructure uses a large wrap-around message queue in a Consoles data space to hold messages before they are queued to a console. This prevents message floods from exhausting console buffer storage but has the side-effect that messages must pass through console queuing in order to be removed. The only mechanism for removing a message from a console is either normal display processing or the K Q command, which must be issued on every console that has a message backlog. And, there is no equivalent of the K Q command to remove unwanted messages from EMCS consoles. Message Flood Automation occurs before messages are sent to the Consoles address space, before the messages are entered into the data space queue, and before messages are queued to consoles -- and can prevent all of these things from happening.

Message Flood Automation and EMCS consoles

Message Flood Automation processing occurs before a message is queued to Extended MCS (EMCS) consoles. Because EMCS console interfaces may be used by automation products such as NetView, there are special considerations:

- If a message flood occurs, and Message Flood Automation has been requested to suppress the message from display, the message will not be queued to any EMCS console unless automation of the message has been requested (typically by specifying AUTO on the message's MPFLSTxx entry).
- If a message flood occurs, and Message Flood Automation has been requested to suppress the message from display and NOT automate it, the message will not be queued to any EMCS console.

Note that the decisions to log, display or automate a message are independent decisions. It is possible

(and may be desirable) to obtain messages at EMCS consoles for automation purposes without logging or displaying them.

If the NetView MSGIFAC parameter is set to SYSTEM, NetView will obtain all messages for automation processing from the EMCS console interfaces. (If the MSGIFAC parameter is set to SSIEXT, NetView will only obtain command response messages from the EMCS console interfaces; unsolicited messages are obtained from the SSI). The messages that NetView "sees" at the EMCS console interface have already been processed by both Message Flood Automation and z/OS deletion, logging and console queuing processing.

Limitations

Message Flood Automation has the following limitations (where the names in parentheses are the names of the specific MPF exit flags that are tested for):

- It only sees messages for which a specific MPF exit does NOT exist. Messages for which a USEREXIT specification exists in MPFLSTxx will NOT be seen by Message Flood Automation. This includes USEREXIT specifications associated with generic message ID statements and USEREXIT specifications associated with .DEFAULT statements.
- It only counts the first (major) line of multi-line messages.
- It does not handle branch-entry messages until they are re-issued as normal messages. It cannot affect them while they are being queued for re-issue.
- It specifically ignores IEF196I and IEF170I messages.
 - IEF170I
A write-to-programmer message operation failed. The IEF170I message includes the reason for the failure and 53 characters of the failing message's text.
 - IEF196I
A message from a task started under the Master Subsystem (MSTR) is being written to the system log because it could not be written to the system message data set or joblog data set. The IEF196I message includes the message ID and text of the failing message.
- It specifically ignores WTOR messages (CTXTTFWR).

If Message Flood Automation does not take action against a message, it is often because of these restrictions.

Operator Commands

Operator commands exist to do the following:

- Enable message flood checking
- Disable message flood checking
- Re-initialize the counts, indicators and actions, and read the specified MSGFLDxx PARMLIB

member

- Display the status of the Message Flood Automation function
- Display whether intensive mode is active for the different classes of messages
- Display the counts and parameters used by Message Flood Automation
- Modify the counts and parameters used by Message Flood Automation
- Enable message rate information gathering
- Disable message rate information gathering
- Display message rate information
- Free the common storage area that is used by Message Flood Automation

All of these commands are implemented through a formal z/OS Command Exit.

Message Flood Automation **SET**, **SETMF** and **DISPLAY** commands do not perform authorization checks and therefore cannot be restricted through the installation's security product.

Because the Message Flood Automation **SET** command is not implemented as part of the operating system **SET** command processor, you cannot combine the Message Flood Automation **SET** command with other operating system **SET** commands. (The operating system **SET** command allows multiple **SET** commands to be combined into one; the Message Flood Automation **SET** command does not). The Message Flood Automation **SET** command must be specified by itself.

Re-initialize the Message Flood Automation parameters

Re-initialize the control variables, counts, and indicators to the default values set by Message Flood Automation itself and then read the requested MSGFLDxx PARMLIB member. The operation therefore nullifies the effect of any values set previously by command. Re-initialization requires that the common storage area has been obtained but does not require that message flood checking be enabled. The common storage area is used to hold the Message Flood Automation parameters and counters and is used for communications between the message processing and command processing parts of Message Flood Automation.

```
SET MSGFLD=xx
```

```
T MSGFLD=xx
```

where *xx* is the suffix of a MSGFLDxx PARMLIB member. The suffix is limited to the alphanumeric characters A-Z and 0-9. National and special characters are not supported.

PLEASE NOTE: The actual reading of the PARMLIB member will occur in the DUMPSRV address space. PARMLIB reading is performed in the DUMPSRV address space because the Console address space -- where Message Flood Automation command processing occurs -- does not support the dynamic allocation services required by the PARMLIB reading service.

Message **CNZZ016I** is issued when the Message Flood Automation parameters have been successfully re-initialized. Message **CNZZ401I** is issued when Message Flood Automation attempts to read the requested PARMLIB member. Message **CNZZ410I** is issued when reading of the PARMLIB member is complete.

Information about the active PARMLIB member is provided on the **CNZZ042I** multi-line message in response to a **DISPLAY MSGFLD,STATUS** command.

Do not place a SET MSGFLD=xx command in a COMMNDxx member of PARMLIB to automatically load a Message Flood Automation PARMLIB member. COMMNDxx processing occurs before any system address spaces capable of supporting dynamic allocation become active. The system service that Message Flood Automation uses to read PARMLIB requires dynamic allocation. Any attempt to read the MSGFLDxx PARMLIB member prior to the availability of the DUMPSRV address space will be ignored.

Enable message flood checking

Enable message flood checking. The enable switch is set on and processing resumes. Messages received while message flood checking is disabled are not processed i.e. they are not counted.

```
SETMF ON
```

Message **CNZZ041I** is issued in response to the **SETMF ON** command.

Information about the Message Flood Automation enablement state is provided on the **CNZZ042I** multi-line message in response to a **DISPLAY MSGFLD,STATUS** command.

The **SETMF ON** command cannot be placed in the COMMNDxx member of PARMLIB to enable Message Flood Automation processing because COMMNDxx processing occurs before the CNZZCMXT command exit is automatically loaded by the system during IPL.

Disable message flood checking

Disable message flood checking. No values are changed and when subsequently enabled, processing resumes with the values as set at the time of the disablement command.

```
SETMF OFF
```

Message **CNZZ041I** is issued in response to the **SETMF OFF** command.

Information about the Message Flood Automation enablement state is provided on the **CNZZ042I** multi-line message in response to a **DISPLAY MSGFLD,STATUS** command.

Display individual Message Flood Automation parameters

Display the value of one or all parameters.

```
DISPLAY MSGFLD,MSGTYPE=msgtype,keyword
```

```
DISPLAY MF,MSGTYPE=msgtype,keyword
```

D **MSGFLD**, MSGTYPE=msgtype, keyword

D **MF**, MSGTYPE=msgtype, keyword

where *msgtype* specifies the message type processing for which the values are to be displayed. Valid *msgtype* specifications are: **REGULAR**, **ACTION** and **SPECIFIC**.

Valid *keyword* specifications are: **MSGTHRESH**, **JOBTHRESH**, **INTVLTIME**, **SYSIMTIME**, **JOBIMTIME**, **MSGLIMIT** and **MSGIMTIME**.

The following combinations of *msgtype* and *keyword* are supported:

Table 1. msgtype and keyword combinations

MSGTYPE=REGULAR	MSGTYPE=ACTION	MSGTYPE=SPECIFIC
INTVLTIME	INTVLTIME	INTVLTIME
JOBIMTIME	JOBIMTIME	
JOBTHRESH	JOBTHRESH	
		MSGIMTIME
		MSGLIMIT
MSGTHRESH	MSGTHRESH	MSGTHRESH
SYSIMTIME	SYSIMTIME	SYSIMTIME

Example:

```
DISPLAY MSGFLD,MSGTYPE=REGULAR, JOBIMTIME
```

The **CNZZ301I** message is issued in response to the **DISPLAY MSGFLD,MSGTYPE=msgtype,keyword** command.

Display all of the Message Flood Automation parameters

The **PARAMETERS** keyword may be specified to display the current values of all of the parameters for all of the *msgtypes*.

```
DISPLAY MSGFLD, PARAMETERS
```

```
DISPLAY MF, PARAMETERS
```

```
D MSGFLD, PARAMETERS
```

```
D MF, PARAMETERS
```

The **CNZZ901I** multi-line message is issued in response to the **DISPLAY MSGFLD,PARAMETERS** command.

Display all of the Message Flood Automation default actions

The **DEFAULTS** keyword may be specified to display the current *default* actions to be taken for all of the *msgtypes*.

```
DISPLAY MSGFLD,DEFAULTS
```

```
DISPLAY MF,DEFAULTS
```

```
D MSGFLD,DEFAULTS
```

```
D MF,DEFAULTS
```

The values displayed are the Message Flood Automation built-in defaults as modified by the actions specified on the **DEFAULT** statements in the active **MSGFLDxx PARMLIB** member.

The **CNZZ904I** multi-line message is issued in response to the **DISPLAY MSGFLD,DEFAULTS** command.

Display the default job actions for REGULAR and ACTION messages

The **JOBS** keyword may be specified to display the current *default* actions to be taken for all of the *jobs* that have been defined in the active **MSGFLDxx PARMLIB** member.

```
DISPLAY MSGFLD,JOBS
```

```
DISPLAY MF,JOBS
```

```
D MSGFLD,JOBS
```

```
D MF,JOBS
```

The values displayed are the Message Flood Automation built-in defaults as modified by the actions specified on the **JOB** statements in the active **MSGFLDxx PARMLIB** member.

The **CNZZ905I** multi-line message is issued in response to the **DISPLAY MSGFLD,JOBS** command.

Display the default actions for SPECIFIC messages

The **MSGS** keyword may be specified to display the current *default* actions to be taken for all of the *messages* that have been defined in the active **MSGFLDxx PARMLIB** member.

```
DISPLAY MSGFLD,MSGS
```

```
DISPLAY MF,MSGS
```

```
D MSGFLD,MSGS
```

```
D MF,MSGS
```

The values displayed are the Message Flood Automation built-in defaults as modified by the actions specified on the MSG statements in the active MSGFLDxx PARMLIB member.

The **CNZZ906I** multi-line message is issued in response to the **DISPLAY MSGFLD,MSG** command.

Display the Message Flood Automation status

The **STATUS** keyword may be specified to display the current enablement status of Message Flood Automation as well as the name of the currently active MSGFLDxx PARMLIB member.

```
DISPLAY MSGFLD, STATUS
```

```
DISPLAY MF, STATUS
```

```
D MSGFLD, STATUS
```

```
D MF, STATUS
```

The **CNZZ042I** multi-line message is issued in response to the **DISPLAY MSGFLD,STATUS** command.

Display the Message Flood Automation intensive mode states

The **MODE** keyword may be specified to display the current intensive mode states for the three message types.

```
DISPLAY MSGFLD, MODE
```

```
DISPLAY MF, MODE
```

```
D MSGFLD, MODE
```

```
D MF, MODE
```

The **CNZZ040I** message is issued in response to the **DISPLAY MSGFLD,MODE** command.

Information about intensive mode states is also provided on **CNZZ042I** multi-line message in response to a **DISPLAY MSGFLD,STATUS** command.

Modify individual Message Flood Automation parameters

Modify the Message Flood Automation parameters being used.

```
SETMF MSGTYPE=msgtype,  
      keyword=value [, keyword=value]
```

The **SETMF** command uses the same *msgtype* and *keyword* specifications as the **DISPLAY MSGFLD** command, so any value that may be displayed by **DISPLAY MSGFLD** may be set using the **SETMF** command.

One or more *keyword=value* pairs may be specified, separated by a comma.

Example:

```
SETMF MSGTYPE=REGULAR, JOBIMTIME=10, SYSIMTIME=10
```

The specification is checked for syntax. If there is any error, a message is issued and no values are updated.

The SETMF command is used to alter the common storage area copy of the Message Flood Automation parameters. If you wish the change to be permanent, you must update the parameter in a MSGFLDxx PARMLIB member. Changes made by SETMF persist only until the next SET MSGFLD= command or IPL.

The **CNZZ208I** message is produced in response to the **SETMF MSGTYPE=msgtype,keyword=value** command.

Modify the Message Flood Automation default actions

Modify the default actions being used by Message Flood Automation.

```
SETMF MSGTYPE=msgtype,  
      DEFAULT=action[,action]
```

The SETMF DEFAULT command supports the same *msgtype* and *action* specifications as the DEFAULT PARMLIB statement, so any action that may be specified by the DEFAULT PARMLIB statement may be set using the SETMF DEFAULT command.

One or more *actions* may be specified, separated by a comma.

Example:

```
SETMF MSGTYPE=SPECIFIC, DEFAULT=NOAUTO, NODISPLAY
```

The specification is checked for syntax. If there is any error, a message is issued and no actions are updated.

The SETMF DEFAULT command is used to alter the common storage area copy of the Message Flood Automation default actions. If you wish the change to be permanent, you must update the action in a MSGFLDxx PARMLIB member. Changes made by SETMF DEFAULT persist only until the next SET MSGFLD= command or IPL.

The **CNZZ208I** message is produced in response to the **SETMF MSGTYPE=msgtype,DEFAULT=action** command.

Modify the job actions for REGULAR and ACTION messages

Modify the actions that will be taken by Message Flood Automation against jobs producing REGULAR and ACTION messages.

```
SETMF MSGTYPE=msgtype,  
      JOB=jobname  
      [,action] [,action]
```

The SETMF JOB command supports the same *msgtype* and *action* specifications as the JOB PARMLIB statement, so any action that may be specified by the JOB PARMLIB statement may be set using the SETMF JOB command.

One or more *actions* may be specified, separated by a comma.

Example:

```
SETMF MSGTYPE=REGULAR, JOB=REGJOB08, NOAUTO, NODISPLAY
```

The specification is checked for syntax. If there is any error, a message is issued and no actions are updated.

You can use the **SETMF MSGTYPE=msgtype, JOB=jobname** command to define a *new* job that you want to define specific actions for. If the jobname that you specify is not one that Message Flood Automation recognizes, and space is available in the jobname table, then Message Flood Automation will add the jobname to the jobname table and take the specified actions should that job produce a message flooding situation. Please note that there is no way to remove a jobname from the jobname table except by loading a MSGFLDxx PARMLIB member.

The SETMF JOB command is used to alter the common storage area copy of the Message Flood Automation actions. If you wish the change to be permanent, you must update the action in a MSGFLDxx PARMLIB member. Changes made by SETMF JOB persist only until the next SET MSGFLD= command or IPL.

The **CNZZ208I** message is produced in response to the **SETMF MSGTYPE=msgtype, JOB=jobname, ACTIONS=** command.

Modify the actions for SPECIFIC messages

Modify the actions that Message Flood Automation will take for SPECIFIC messages.

```
SETMF MSGTYPE=SPECIFIC,  
      MSG=msgid  
      [,action] [,action]
```

The SETMF MSG command supports the same *msgtype* and *action* specifications as the MSG PARMLIB statement, so any action that may be specified by the MSG PARMLIB statement may be set using the SETMF MSG command.

One or more *actions* may be specified, separated by a comma.

Example:

```
SETMF MSGTYPE=SPECIFIC, MSG=IEA000I, NOAUTO, NODISPLAY
```

The specification is checked for syntax. If there is any error, a message is issued and no actions are updated.

You can use the **SETMF MSGTYPE=SPECIFIC,MSG=msgid** command to define a *new* message that you want to define specific actions for. If the msgid that you specify is not one that Message Flood Automation recognizes, and space is available in the msgid table, then Message Flood Automation will add the msgid to the msgid table and take the specified actions should that message be involved in a message flooding situation. Please note that there is no way to remove a msgid from the msgid table except by loading a MSGFLDxx PARMLIB member.

The SETMF MSG command is used to alter the common storage area copy of the Message Flood Automation message actions. If you wish the change to be permanent, you must update the action in a MSGFLDxx PARMLIB member. Changes made by SETMF MSG persist only until the next SET MSGFLD= command or IPL.

The **CNZZ208I** message is produced in response to the **SETMF MSGTYPE=msgtype,MSG=msgid** command.

Release the Message Flood Automation common storage (SQA) area

Release (FREEMAIN) the common storage (SQA) area being used. Release requires that message flood checking and message rate monitoring be disabled.

SETMF FREE

The common storage area is used to hold the Message Flood Automation parameters and counters and is used for communication between the message processing and command processing parts of Message Flood Automation. The common storage used resides in the operating system's System Queue Area (SQA).

You should issue this command only if you wish to:

- install an updated level of the Message Flood Automation code
- completely disable Message Flood Automation and remove it from the system

Once the common storage area has been freed, it cannot be re-acquired without recycling IEAVMXIT by issuing a K M,UEXIT=Y command.

The **CNZZ026I** message is issued in response to the **SETMF FREE** command.

Messages **CNZZ027I** or **CNZZ028I** may be issued if errors occur why attempting the operation.

Enable message rate monitoring

Enable the collection of message rate information.

SETMF MONITORON

This command can also be used to re-initialize the message rate information.

It is recommended that you gather message rate information for at least an hour before displaying it. In general, the longer the sample, the more accurate the results. Message rate monitoring incurs more

overhead than standard (non-intensive mode) message flood processing and should probably not be run all of the time. A 24-hour sample taken during a busy period every few months is probably sufficient.

NOTE: Message rate monitoring measures only the messages that Message Flood Automation can do something about. It is subject to the same limitations as Message Flood Automation (see the Limitations section).

The largest interval between successive messages that will be recorded by Message Rate Monitoring is 2048 seconds, or approximately 33 minutes. The smallest interval that can be recorded is 250 picoseconds.

NOTE: Message rate monitoring is turned off and the counters are re-initialized whenever a SET MSGFLD=xx command is issued.

The **CNZZ902I** message is issued in response to the **SETMF MONITORON** command.

Disable message rate monitoring

Disable the collection of message rate information.

```
SETMF MONITOROFF
```

Disabling the collection of message rate information does not re-initialize the information already collected.

The **CNZZ903I** message is issued in response to the **SETMF MONITOROFF** command.

Display message rate information

Display message rate information collected by the message rate monitoring process.

```
DISPLAY MSGFLD,MSGRATE [,n]
```

```
DISPLAY MF,MSGRATE [,n]
```

```
D MSGFLD,MSGRATE [,n]
```

```
D MF,MSGRATE [,n]
```

where *n* is an optional graph length parameter. The default is 25. Note that the command processor will adjust this value to obtain the best scaling. The smallest supported graph has a length of 8; the largest supported graph has a length of 200. Standard graph sizes are: 8, 10, 16, 20, 25, 32, 40, 50, 80, 100 and 200.

The message rate information gathered is presented in multi-line message **CNZZ043I**:

- the total number of messages counted
- the total elapse time from when message rate monitoring was started to the current time
- the average message rate (in messages / second) from when message rate monitoring was started

to the current time

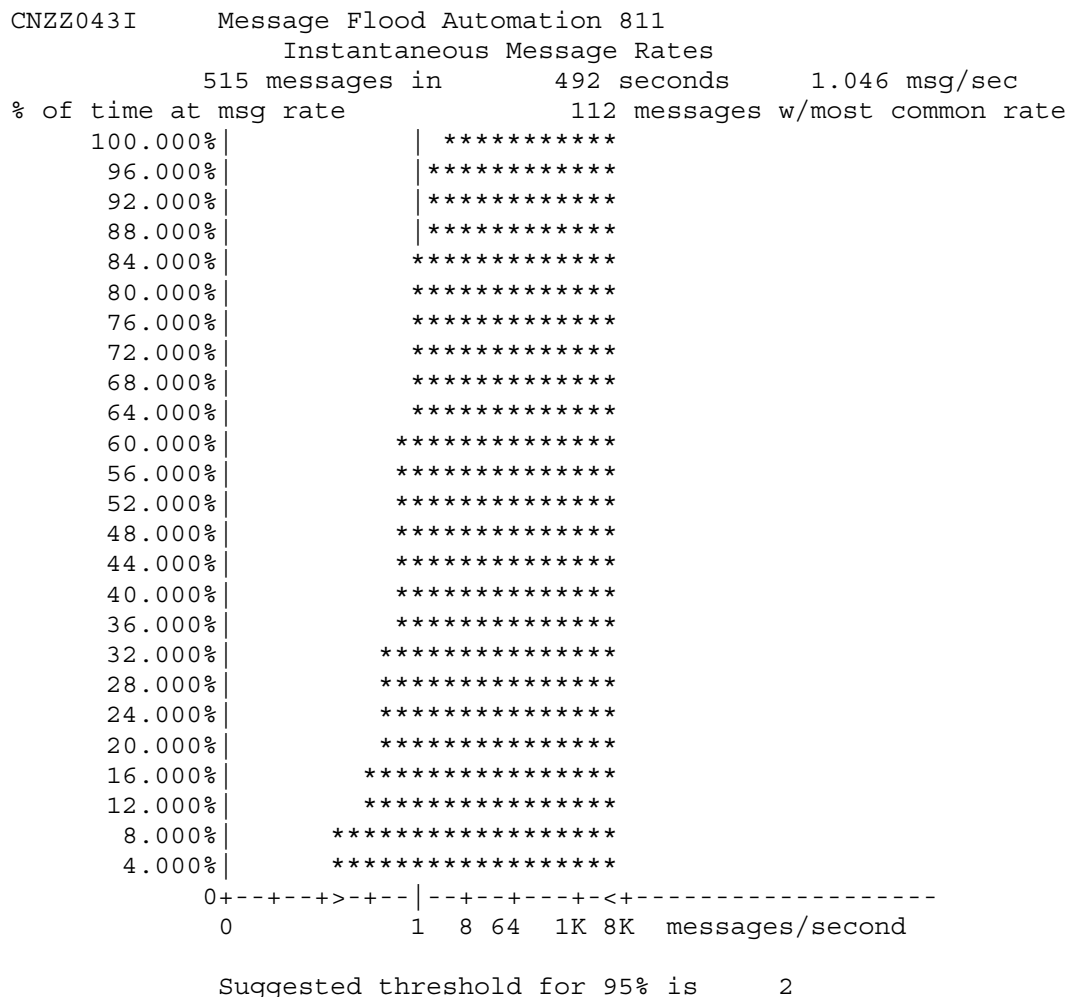
- the number of messages occurring at the most common message rate
- a message rate distribution graph

The message rate distribution graph shows the percent of *time* at a given message rate on the Y-axis and instantaneous message rates (in messages / second) on the X-axis. The X-axis scale is logarithmic with each character position being a factor of 2 greater than the previous position in the rightward direction. Tickmarks are provided at 8X intervals.

Each vertical bar of asterisks in the graph is rightward cumulative, that is, each bar represents not only the fraction of time at its own rate, but the fraction of time with a lesser rate. (A bar's own contribution to the time at a given message rate is therefore the difference between its height and the height of its immediate leftward neighbor).

A vertical line (|) indicates the most common message rate.

The graph should have a characteristic "S" shape to it caused by there being relatively few messages occurring at very low message rates (the bottom left of the S curve) and very few messages occurring at very high message rates (the top right of the S curve).



```

Suggested threshold for 96% is      3
Suggested threshold for 97% is      3
Suggested threshold for 98% is      4
Suggested threshold for 99% is      6

```

This example was produced using a testcase that issued messages with an exponential distribution of inter-arrival times and a mean inter-arrival time of 0.5 seconds. The vertical bar indicates that the most common (mean) message rate is 1 messages / second. The average message rate is only slightly more than 1 message / second, a rate that has been determined by IBM human factor studies to be the maximum rate that messages should be presented on any one console.

On the X-axis, the minimum and maximum message rates recorded are indicated (by the > and < symbols respectively) on either side of the mean message rate. The percentage of messages occurring at the maximum message rate is usually quite small and may not be visible unless the resolution of the graph is improved by increasing the number of message lines in the graph.

The graph presents instantaneous message rates that are determined from the inter-arrival times of the messages. Small inter-arrival times result in high instantaneous message rates; large inter-arrival times result in low instantaneous message rates. A high message rate on the graph does not necessarily imply that multiple, consecutive messages were issued at that rate. It is quite possible (as in the example) for a high message rate to be indicated without Message Flood Automation being triggered. (It is multiple, consecutive, high message rate messages that trigger Message Flood Automation).

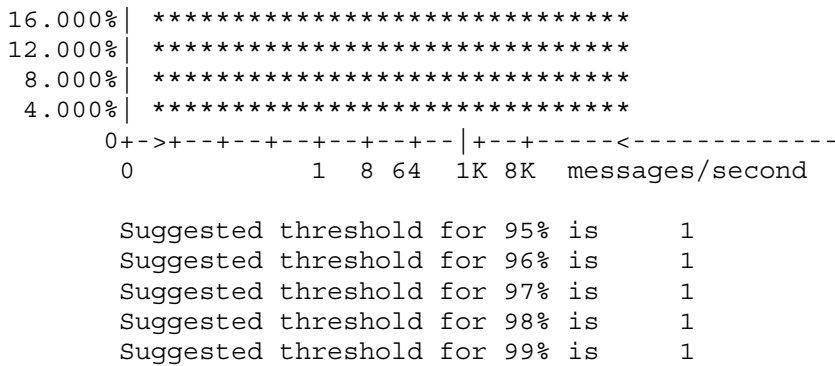
The suggested threshold values represent the message rates that are not exceeded some fraction of the time. In the example, a message rate of 4 messages / second is not exceeded 98% of the time; a message rate of 6 messages / second is not exceeded 99% of the time. You can use the suggested threshold values to set an appropriate REGULAR MSGTHRESH value.

Let's look at a more interesting graph:

```

CNZZ043I      Message Flood Automation
              Instantaneous Message Rates
              34299 messages in 78111 seconds      0.439 msg/sec
% of time at msg rate      5993 messages w/most common rate
100.000% | *****
 96.000% | *****
 92.000% | *****
 88.000% | *****
 84.000% | *****
 80.000% | *****
 76.000% | *****
 72.000% | *****
 68.000% | *****
 64.000% | *****
 60.000% | *****
 56.000% | *****
 52.000% | *****
 48.000% | *****
 44.000% | *****
 40.000% | *****
 36.000% | *****
 32.000% | *****
 28.000% | *****
 24.000% | *****
 20.000% | *****

```



This graph looks very different from the previous one. The first reaction of most people is to look at the average message rate of 0.439 messages per second and the fact that the most commonly occurring message has a rate of 512 messages per second(!) and wonder how these two statistics can be reconciled. It is important to understand what an average can tell you and what it cannot. What an average can tell you is that (in this case) 34299 messages occurred during the 78111 second interval that was monitored. **What the average message rate cannot tell you is *how* those messages were distributed within the monitoring interval.** If the messages were distributed uniformly within the monitoring interval, *the time between messages would be the same* -- but a quick look at the graph shows this to not be the case: there were some number of messages that occurred at an instantaneous message rate of 1 message every 1024 seconds (at the left edge of the graph) and there were some number of messages that occurred at an instantaneous message rate of 262144 messages per second (at the right edge of the graph). And there were the 5993 "most commonly occurring" messages that occurred at a rate of 512 messages per second. The answer to this riddle is that one or more message "spikes" occurred at some point in the monitoring interval, and those spikes produced at least 5993 messages at a rate of 512 messages per second. Why doesn't this very high message rate affect the overall average message rate? Because, this very high message rate only occurred for 11.7 seconds (5993/512) -- which represents only 0.015% of the time within the interval of 78111 seconds.

The very broad "top" to the graph is indicative of a very small number of messages that occurred with very high instantaneous message rates. However, these messages occur for such brief periods of time that they have almost no effect on the overall message rate. The very broad "base" of the graph is indicative of a very non-uniform distribution of messages within the monitoring interval.

The "suggested thresholds" are all one because one is the lowest value that can be specified.

Setting thresholds based on message rates

Message Flood Automation thresholds should be set based on the mean (most common) message rate, not on the maximum message rates.

The Message Rate Monitoring function measures the message rate for all messages that are subject to control by Message Flood Automation. The suggested thresholds provided in message CNZZ043I in response to a DISPLAY MSGFLD,MSGRATE command are good values to start with.

- You should set your REGULAR message threshold (MSGTHRESH) somewhat *higher* than the mean message rate and your REGULAR message inter-message time (SYSIMTIME) *at or slightly below* the mean message rate inter-message time. Note that inter-message time is the inverse of message rate: an average message rate of 2.0 messages / second means that messages arrive on average every 0.5 seconds (so the inter-message time is 0.5 seconds).

- The REGULAR JOB message threshold (JOBTHRESH) **must** be set to a value less than that of MSGTHRESH. A JOBTHRESH value that is 30-40% of MSGTHRESH is a good starting point. This will allow you to handle 2-3 message flooding jobs simultaneously. A general "Rule of Thumb" is to take the MSGTHRESH value and divide by the number of jobs (less than 10) that you want Message Flood Automation to be able to handle simultaneously and use the result as the JOBTHRESH value.
- The message rate for ACTION messages is typically a small fraction of REGULAR messages, so your ACTION message threshold (MSGTHRESH) can be *less* than your REGULAR message threshold. (Setting the ACTION threshold lower than the REGULAR threshold does not increase your overhead because the ACTION messages occur less frequently.) Because the ACTION message rate is lower, the ACTION inter-message time (SYSIMTIME) can be *greater* than your REGULAR message inter-message time.
 - The ACTION JOB message threshold (JOBTHRESH) **must** be set to a value less than that of MSGTHRESH. A JOBTHRESH value that is 30-40% of MSGTHRESH is a good starting point. This will allow you to handle 2-3 message flooding jobs simultaneously. A general "Rule of Thumb" is to take the MSGTHRESH value and divide by the number of jobs (less than 10) that you want Message Flood Automation to be able to handle simultaneously and use the result as the JOBTHRESH value.
- Unless you have chosen very common messages, the message rate for SPECIFIC messages is typically a small fraction of REGULAR messages, so your SPECIFIC message threshold (MSGTHRESH) can be *less* than your REGULAR message threshold. Because the SPECIFIC message rate is lower, the SPECIFIC inter-message time (SYSIMTIME) can be *greater* than your REGULAR message inter-message times.
 - The SPECIFIC MSG message threshold (MSGLIMIT) **must** be set to a value less than that of MSGTHRESH. A MSGLIMIT value that is 15-20% of MSGTHRESH is a good starting point. This will allow you to handle 5-6 SPECIFIC message flooding messages simultaneously. A general "Rule of Thumb" is to take the MSGTHRESH value and divide by the number of messages (less than 30) that you want Message Flood Automation to be able to handle simultaneously and use the result as the MSGLIMIT value.

The message rate specified by a threshold is also a function of the interval over which the threshold number of messages occurs. You can specify the same message rate through different combinations of the threshold and interval values. For example, setting MSGTHRESH=50 and INTVLTIME=1 specifies a message rate of 50 messages / second. Setting MSGTHRESH=100 and INTVLTIME=2 also specifies a message rate of 50 messages / second. You may wish to choose which way you specify the message rates to achieve other goals:

- Using the MSGTHRESH=50 and INTVLTIME=1 specification will make Message Flood Automation more responsive to detecting message flooding situations because only 50 messages will be counted between computations of the message rate; however, the overhead of the message rate computation will be incurred twice as frequently as the MSGTHRESH=100 and INTVLTIME=2 specification.
- Using the MSGTHRESH=100 and INTVLTIME=2 specification will make Message Flood Automation less responsive to detecting message flooding situations because 100 messages will be counted between computations of the message rate; however, the overhead of the message rate

computation will be incurred half as frequently as the MSGTHRESH=50 and INTVLTIME=1 specification.

You can use different combinations of threshold and interval to trade-off message flood detection responsiveness and message flood detection overhead.

The general idea is to set the various thresholds high enough that they are not being triggered by normal fluctuations in message rates but are triggered when sudden, very high message rates are encountered. For REGULAR messages, using one of the suggested threshold values provided by the CNZZ043I message is a good first approximation. You should set your thresholds high enough that Message Flood Automation is not constantly oscillating into and out of intensive mode. Receiving message CNZZ001I is usually a good indication that you have set the REGULAR message threshold too low; receiving message CNZZ019I is usually a good indication that you have set the ACTION message threshold too low.

Message flood detection behavior

Message Flood Automation processing is driven by its three message class message counters. Each counter counts MSGTHRESH messages and is then reset to zero, to begin counting again. When each counter counts its first message (of a new interval), it stores a timestamp to indicate when the interval began, and when MSGTHRESH messages have been counted, another timestamp indicating when the interval ended. If the difference between the starting and ending timestamps is less than INTVLTIME, then intensive mode is entered.

A message flood can begin when a message counter is zero, one, equal to MSGTHRESH -- or anywhere inbetween.

- If the message counter is zero, the first message of the flood will cause the timestamp to be stored and MSGTHRESH messages later (assuming this occurs in less than INTVLTIME), cause the threshold exceeded message to be issued and intensive mode to be entered. For this case, only MSGTHRESH messages are required to enter intensive mode.
- If the message counter is one, the timestamp marking the beginning of the interval *has already been stored*, and after MSGTHRESH-1 messages have been counted and the ending timestamp acquired, *the difference between the timestamps may not cause intensive mode to be entered*. If a flood is underway, the next MSGTHRESH number of flood messages will cause intensive mode to be entered. In this case, it will take $2 \times \text{MSGTHRESH} - 1$ flood messages to cause intensive mode to be entered.
- If the message counter is already at MSGTHRESH, the first flood message will cause the ending timestamp to be stored, and the difference in timestamps will probably cause intensive mode to not be entered. However, the next MSGTHRESH flood messages will cause intensive mode to be entered. So in this case, it will take $\text{MSGTHRESH} + 1$ messages to cause intensive mode to be entered.
- If the message counter is between one and MSGTHRESH, it will take $2 \times \text{MSGTHRESH} - n$ messages to cause intensive mode to be entered, where "n" is the number of messages already counted.

The bottom line is that the triggering of intensive mode may not occur precisely after MSGTHRESH

flood messages have occurred.

Command Summary

DISPLAY MSGFLD command

```
DISPLAY MSGFLD, PARAMETERS
DISPLAY MSGFLD, DEFAULTS
DISPLAY MSGFLD, JOBS
DISPLAY MSGFLD, MSGS
```

```
DISPLAY MSGFLD, MODE
DISPLAY MSGFLD, MSGRATE [, n]
DISPLAY MSGFLD, STATUS
```

```
DISPLAY MSGFLD, MSGTYPE=ACTION { , JOBTHRESH }
                                { , MSGTHRESH }
                                { , INTVLTIME }
                                { , JOBIMTIME }
                                { , SYSIMTIME }
```

```
DISPLAY MSGFLD, MSGTYPE=REGULAR { , JOBTHRESH }
                                { , MSGTHRESH }
                                { , INTVLTIME }
                                { , JOBIMTIME }
                                { , SYSIMTIME }
```

```
DISPLAY MSGFLD, MSGTYPE=SPECIFIC { , MSGTHRESH }
                                { , INTVLTIME }
                                { , SYSIMTIME }
                                { , MSGIMTIME }
                                { , MSGLIMIT }
```

DISPLAY may be abbreviated **D** and **MSGFLD** may be abbreviated **MF**.

SET MSGFLD command

```
SET MSGFLD=xx
```

SET may be abbreviated **T**.

SETMF command

```
SETMF ON
SETMF OFF
SETMF FREE
```

```
SETMF MONITORON
SETMF MONITOROFF
```

```
SETMF MSGTYPE=ACTION { , JOBTHRESH=value }
                     { , MSGTHRESH=value }
                     { , INTVLTIME=value }
                     { , JOBIMTIME=value }
```

```
{,SYSIMTIME=value}
```

where *value* is a count of messages, or time in seconds or fractions of a second (SYSIMTIME and JOBIMTIME only)

```
SETMF MSGTYPE=ACTION,  
      DEFAULT=action[,action]
```

```
SETMF MSGTYPE=ACTION, JOB=jobname,  
      [,action] [,action]
```

where *action* is LOG|NOLOG, DISPLAY|NODISPLAY,
AUTO|NOAUTO, RETAIN|NORETAIN,
CMD|NOCMD

```
SETMF MSGTYPE=REGULAR{,JOBTHRESH=value}  
      {,MSGTHRESH=value}  
      {,INTVLTIME=value}  
      {,JOBIMTIME=value}  
      {,SYSIMTIME=value}
```

where *value* is a count of messages, or time in seconds or fractions of a second (SYSIMTIME and JOBIMTIME only)

```
SETMF MSGTYPE=REGULAR,  
      DEFAULT=action[,action]
```

```
SETMF MSGTYPE=REGULAR, JOB=jobname  
      [,action] [,action]
```

```
SETMF MSGTYPE=SPECIFIC{,MSGTHRESH=value}  
      {,INTVLTIME=value}  
      {,SYSIMTIME=value}  
      {,MSGIMTIME=value}  
      {,MSGLIMIT=value}
```

where *value* is a count of messages, or time in seconds or fractions of a second (SYSIMTIME and MSGIMTIME only)

```
SETMF MSGTYPE=SPECIFIC,  
      DEFAULT=action[,action]
```

```
SETMF MSGTYPE=SPECIFIC,MSG=msgid  
      [,action] [,action]
```

Built-in defaults

If Message Flood Automation is enabled using a **SETMF ON** command before a MSGFLDxx PARMLIB member has been read using a **SET MSGFLD=xx** command, the following built-in defaults are used:

- REGULAR MSGTHRESH=50
- REGULAR JOBTHRESH=20
- REGULAR INTVLTIME=1

- REGULAR SYSIMTIME=2
- REGULAR JOBIMTIME=2

REGULAR actions: LOG, AUTO, NODISPLAY, NOCMD

- ACTION MSGTHRESH=50
- ACTION JOBTHRESH=20
- ACTION INTVLTIME=1
- ACTION SYSIMTIME=2
- ACTION JOBIMTIME=2

ACTION actions: LOG, AUTO, NODISPLAY, NOCMD, NORETAIN

- SPECIFIC MSGTHRESH=50
- SPECIFIC MSGLIMIT=20
- SPECIFIC INTVLTIME=1
- SPECIFIC SYSIMTIME=2
- SPECIFIC MSGIMTIME=2

SPECIFIC actions: LOG, AUTO, NODISPLAY, NORETAIN, NOIGNORE

The following SPECIFIC messages are built-in:

- IOS000I NOLOG,NOAUTO,NODISPLAY,NORETAIN,NOIGNORE
- IOS002A NOLOG,NOAUTO,NODISPLAY,NORETAIN,NOIGNORE
- IOS017I NOLOG,NOAUTO,NODISPLAY,NORETAIN,NOIGNORE
- IOS107I NOLOG,NOAUTO,NODISPLAY,NORETAIN,NOIGNORE
- IOS109E NOLOG,NOAUTO,NODISPLAY,NORETAIN,NOIGNORE
- IOS251I NOLOG,NOAUTO,NODISPLAY,NORETAIN,NOIGNORE
- IOS291I NOLOG,NOAUTO,NODISPLAY,NORETAIN,NOIGNORE
- IOS444I NOLOG,NOAUTO,NODISPLAY,NORETAIN,NOIGNORE
- IOS450E NOLOG,NOAUTO,NODISPLAY,NORETAIN,NOIGNORE
- IEA476E NOLOG,NOAUTO,NODISPLAY,NORETAIN,NOIGNORE
- IEA491E NOLOG,NOAUTO,NODISPLAY,NORETAIN,NOIGNORE
- IEA494I NOLOG,NOAUTO,NODISPLAY,NORETAIN,NOIGNORE
- IEA497I NOLOG,NOAUTO,NODISPLAY,NORETAIN,NOIGNORE *

This message set was chosen for best operation during GDPS HyperSwap IPL processing which occurs before PARMLIB can be read. * **NOTE:** IEA497I has continuation messages which are issued as separate single line messages, without message IDs, that are not handled.

When a MSGFLDxx PARMLIB member is read, Message Flood Automation first replaces any parameters that it was using with the built-in default values. It then replaces those values with any values specified in the PARMLIB member. This ensures that Message Flood Automation always has valid policy values that it can use. The exception to this are the built-in SPECIFIC messages which are automatically deleted by the first MSGFLDxx PARMLIB member read.

PARMLIB specifications

Installation policy for controlling message flooding situations is specified through the new MSGFLDxx member of PARMLIB.

The following statement types are provided:

- comment statements
- msgtype statements
- DEFAULT statements
- DEFAULTCMD statements
- JOB statements
- MSG statements

All statements must begin in column 1 or they are ignored. (Statements with a blank in column 1 are ignored.)

Statements are 80 columns in length, but only the first 71 columns are processed. Columns 72-80 are ignored. Specifications extending beyond column 71 will be truncated without detection or warning.

Any statement may be repeated if all of its parameters will not fit on a single statement.

- If a parameter is specified on more than one statement, the specification on the last statement is the one that is used.
- If a parameter is specified more than once on a statement, the rightmost specification is the one that is used.
- If mutually-exclusive parameters (such as LOG / NOLOG) are specified on more than one statement, the parameter specified on the last statement is the one that is used.
- If mutually-exclusive parameters are specified on a single statement, the rightmost specification is the one that is used.

Comment text may be specified on any statement but it must be separated from the statement specification by at least one and sometimes two blanks. There is no need to preface the comment text with either an asterisk ("*") or open comment specification ("/*").

comment statements

A *comment* statement is indicated by placing an asterisk ("*") in column 1 or by placing an open comment specification ("/*") in columns 1 and 2. (There is no need to place a closing comment specification on the statement although it is good practice to do so).

msgtype statements

msgtype statements indicate the type of message that the parameters apply to. *msgtype* statements have the following general form:

```
msgtype keyword=value [,keyword=value]
```

where:

- *msgtype* is:
 - **REGULAR**
 - **ACTION**
 - **SPECIFIC**
- *keyword* is:
 - **INTVLTIME**
 - **JOBIMTIME** - not supported for *msgtype* SPECIFIC
 - **JOBTHRESH** - not supported for *msgtype* SPECIFIC
 - **MSGIMTIME** - not supported for *msgtype* REGULAR and ACTION
 - **MSGLIMIT** - not supported for *msgtype* REGULAR and ACTION
 - **MSGTHRESH**
 - **SYSIMTIME**
- *value* for **JOBTHRESH**, **MSGLIMIT** and **MSGTHRESH** is a positive, non-zero integer count of messages in the range 1 to 999999999
- *value* for **INTVLTIME** is a positive, non-zero integer time in *seconds* in the range 1 to 999999999
- *value* for **SYSIMTIME**, **JOBIMTIME** and **MSGIMTIME** is a positive, non-zero *floating point* time in *seconds* in the range 0.000001 to 16777215.0
- **NOTE:** Floating point fractions such as 0.1 may not be stored exactly due to limitations of the floating point format.

NOTE: For the REGULAR and ACTION *msgtypes*, the JOBTHRESH value must be less than the MSGTHRESH value.

NOTE: For the SPECIFIC *msgtype*, the MSGLIMIT value must be less than the MSGTHRESH value.

Syntax rules:

- The *msgtype* specification must begin in column 1 and must be separated from the first keyword by a single blank character.
- The *keyword* and the *value* are separated by an equal (=) character. There must be no blank characters on either side of the equal character.
- *keyword=value* pairs are separated by a comma. There must be no blank characters on either side of the comma.
- If *value* is a floating point number, it does not need to contain a decimal point unless a fractional value is being specified. The specifications 1.0 and 1. and 1 are equivalent as are 0.25 and .25
- Comment text may follow the last *keyword=value* pair. There must be at least one blank character between the *value* and the beginning of the comment text. There is no need to preface the comment text with either an asterisk ("*") or open comment specification ("/*").

Example:

```
REGULAR MSGTHRESH=50, JOBTHRESH=20, INTVLTIME=1 comment
REGULAR SYSIMTIME=0.125, JOBIMTIME=0.25
```

DEFAULT statements

The **DEFAULT** statement specifies the default action to be taken for a specific address space that exceeds the job threshold message rate (**JOBTHRESH / INTVLTIME**) or a specific message that exceeds the message threshold message rate (**MSGLIMIT / INTVLTIME**). The built-in defaults apply otherwise.

The **DEFAULT** statement applies to the previously specified *msgtype*. (Zero or more **DEFAULT** statements should follow each set of *msgtype* statements).

The **DEFAULT** statement has the following general form:

```
DEFAULT action[,action]
```

where *action* is:

- **LOG | NOLOG** - determines whether the message is written to the SYSLOG/OPERLOG or not
- **AUTO | NOAUTO** - determines whether the message is queued (to an EMCS console) for automation or not
- **DISPLAY | NODISPLAY** - determines whether the message is queued to a console for display or not
- **CMD | NOCMD** - determines whether the command defined by the DEFAULTCMD statement is issued or not. Not valid for *msgtype* SPECIFIC.
- **RETAIN | NORETAIN** - determines whether the message is retained by the Action Message Retention Facility (AMRF). Not valid for *msgtype* REGULAR.
- **IGNORE | NOIGNORE** - determines whether Message Flood Automation is to process the message or not. Not valid for *msgtype* ACTION or REGULAR.

Specifying NOLOG, NODISPLAY and NOAUTO together will result in the message being deleted. Deleted messages are not sent to the Console address space or to other systems in the sysplex.

Syntax rules:

- The **DEFAULT** specification must begin in column 1 and must be separated from the first *action* by a single blank character.
- *actions* are separated by a comma. There must be no blank characters on either side of the comma.
- Comment text may follow the last *action*. There must be at least one blank character between the *action* and the beginning of the comment text. There is no need to preface the comment text with either an asterisk ("*") or open comment specification ("/*").

Example:

```
DEFAULT LOG,NOAUTO,NODISPLAY,NOCMD
```

If the **DEFAULT** statement is not specified, the following built-in defaults are applied:

- **REGULAR LOG**, AUTO, NODISPLAY, NOCMD
- **ACTION LOG**, AUTO, NODISPLAY, NOCMD, NORETAIN
- **SPECIFIC LOG**, AUTO, NODISPLAY, NORETAIN, NOIGNORE

DEFAULTCMD statements

The **DEFAULTCMD** statement specifies the default command that will be issued if a **CMD** action has been specified for the address space.

The **DEFAULTCMD** statement applies to the previously specified *msgtype*. (Zero or one **DEFAULTCMD** statements should follow each set of *msgtype* statements).

The **DEFAULTCMD** statement has the following general form:

```
DEFAULTCMD 'cmdchr,cmdtext'
```

where:

- *cmdchr* is a single character that may be used later in the *cmdtext* to indicate where the jobname should be substituted. Only a single substitution is performed.
- *cmdtext* is any valid z/OS or subsystem command. The *cmdtext* is limited to a maximum of 56 characters.

Syntax rules:

- The **DEFAULTCMD** specification must begin in column 1 and must be separated from the opening quote character of the command specification by a single blank character.
- The *cmdchr* character should be chosen so that it does not conflict with any special characters or punctuation characters used in the *cmdtext*.
- The *cmdchr* character must not have blank characters either before it or after it.
- The *cmdchr* is separated from the *cmdtext* by a comma. There must be no blank characters on either side of the comma. Message Flood Automation treats whatever follows the comma as the *cmdtext*.
- The *cmdtext* is any valid z/OS or subsystem command. Please note that there is no practical way for Message Flood Automation to check the correctness of this command, so it is important that the command be specified correctly. An improperly specified command will result in an error when Message Flood Automation attempts to issue it.

The *cmdtext* may include special characters and punctuation characters. If single-quote characters are used as part of the *cmdtext* specification, two single-quote characters must be specified for each desired single-quote character in the command text that will be used.

- Comment text may follow the closing quote character. There must be at least one blank character

between the quote and the beginning of the comment text. There is no need to preface the comment text with either an asterisk ("***") or open comment specification ("*/**").

- The **DEFAULTCMD** specification does not apply to *msgtype* SPECIFIC.

Example:

```
DEFAULTCMD '#,CANCEL #'          z/OS cancel job
DEFAULTCMD '#,*CANCEL,SY1,#'     JES3 cancel job
DEFAULTCMD '#,SEND ''# is issuing a lot of messages'',BRDCST'
```

NOTE: Your security product may prevent you from issuing the command that you have specified on the **DEFAULTCMD** statement. Since the command is issued from the address space that is causing the flood, the address space causing the flood must be authorized to issue the command. In terms of RACF, this means that the address space must have sufficient access authority. To issue the MVS CANCEL command, UPDATE access authority is required.

JOB statements

JOB statements identify up to **10** specific jobs for which specific actions are to be taken if **REGULAR** or **ACTION** messages from the job are involved in a message flooding situation. During intensive mode processing, if the specified job requires action to be taken against it, the actions will be taken from those specified on the **JOB** statement. You can use the **JOB** statement to override the **DEFAULT** (or built-in) actions for this specific job.

The **JOB** statement applies to the previously specified **REGULAR** or **ACTION** *msgtype* statements. (Zero or more **JOB** statements should follow each set of **REGULAR** or **ACTION** *msgtype* statements).

The **JOB** statement has the following general form:

```
JOB jobname action[,action]
```

where:

- *jobname* is 1 to 8 characters in length and has the following forms:

```
TSTABCD
TST*
TST%%T
TST%T*
*
```

- The form 'TSTABCD' indicates a job with that specific name.
 - The name 'NONAME' is the jobname Message Flood Automation assigns to operating system services that issue messages that are not attributable to a particular job. The system assigns the name 'IEESYSAS' to system services that occupy their own address spaces.
- The form 'TST*' indicates a job that has 'TST' as the first three characters, and any other characters following (or none).
- The form 'TST%%T' indicates a job that has the first three characters 'TST', characters

- 4 through 7 may be any value, and the eighth character is 'T' (e.g. a test job).
- The form 'TST%T*' is a hybrid of the second and third types in which the 4th character can be any value and character 6 (and beyond) may be any characters (or none).
- Specifying '*' all by itself matches all jobnames.
- *action*, if specified, is one or more of the actions as defined for the **DEFAULT** statement. If an *action* is not specified, the **DEFAULT** or built-in default actions are used.

Syntax rules:

- The **JOB** specification must begin in column 1 and must be separated from the *jobname* by a single blank character.
- The *jobname* is separated from the first *action*, if specified, by a single blank character. More than one blank character causes any following string to be treated as comment text.
- *actions* are separated by a comma. There must be no blank characters on either side of the comma.
- Comment text may follow the last *action*. There must be at least one blank character between the *action* and the beginning of the comment text. If no *action* is specified, at least *two* blanks must precede the comment text. There is no need to preface the comment text with either an asterisk ("*") or open comment specification ("/*").

Example:

```
JOB PROD1001
JOB PROD1002  comment: note >1 blank required before comment
JOB PROD1003 LOG,NOAUTO
JOB PROD1004 LOG,NOAUTO comment: note only 1 blank required

JOB AOC%NV* LOG,AUTO,RETAIN
JOB LLA* LOG,AUTO
```

MSG statements

The **MSG** statement defines up to **30** specific messages for which specific actions are to be taken if the message is involved in a message flooding situation. A **MSG** statement applies to the previously specified **SPECIFIC** *msgtype* statement. (One or more **MSG** statements should follow the **SPECIFIC** *msgtype* statement).

The **MSG** statement has the following general form:

```
MSG msgid action[,action]
```

where:

- *msgid* is a 1 to 10 character message identifier. Up to the first 10 characters of the first *blank-delimited* token in the message is treated as the message identifier. (The message identifier must be delimited by zero or more leading blanks and at least one trailing blank).

```
IOS494I
```

Note that if the message ID in the message is followed immediately by a special character (before the first blank), the special character is treated as part of the message ID and must be specified for a match to occur. For example:

EC191 :

must be specified to Message Flood Automation exactly as it appears. Some message IDs include embedded "-" characters. Message Flood Automation can handle message IDs with embedded special characters without problem. The key thing to remember is that Message Flood Automation treats everything in the message up to the first blank as the message ID.

The *msgid* identifier does *not* support "wildcard" specifications.

- *action*, if specified, is one or more of the actions as defined for the **DEFAULT** statement. **CMD** is not supported. If an *action* is not specified, the DEFAULT or built-in actions are taken for the message. You can use a MSG statement to override the DEFAULT or built-in actions to be taken for a specific message. If IGNORE is specified, the message is ignored by Message Flood Automation and no action is taken for the message even if the message threshold is exceeded and other actions are specified.

Syntax rules:

- The **MSG** specification must begin in column 1 and must be separated from the *msgid* by a single blank character.
- The *msgid* is separated from the first *action*, if specified, by a single blank character. More than one blank character causes any following string to be treated as comment text.
- *actions* are separated by a comma. There must be no blank characters on either side of the comma.
- Comment text may follow the last *action*. There must be at least one blank character between the *action* and the beginning of the comment text. If no *action* is specified, at least *two* blanks must precede the comment text. There is no need to preface the comment text with either an asterisk ("*") or open comment specification ("/*").

Example:

```
MSG IAT1001
MSG IAT1002  comment: note >1 blank required before comment
MSG IAT1003  LOG,NOAUTO
MSG IAT1004  LOG,NOAUTO comment: note only 1 blank required

MSG IOS000I  NOLOG,NODISPLAY
MSG DSI064A  NOLOG,NODISPLAY,NORETAIN
```

You may wish to provide MSG specifications for the following messages which can be produced if an ESCON Director or FICON switch fails:

- IOS001E
- IOS050I
- IOS051I

- IOS071I
- IOS251I
- IOS444I
- IOS450E

You may wish to provide MSG specifications for the following messages which can be produced if a device fails:

- IOS003A

You may wish to provide MSG specifications for the following messages if you perform HyperSwaps or are in a GDPS environment:

- IOS000I NOLOG,NOAUTO,NODISPLAY,NOCMD
- IOS002A NOLOG,NOAUTO,NODISPLAY,NOCMD
- IOS017I NOLOG,NOAUTO,NODISPLAY,NOCMD
- IOS107I NOLOG,NOAUTO,NODISPLAY,NOCMD
- IOS109E NOLOG,NOAUTO,NODISPLAY,NOCMD
- IOS251I NOLOG,NOAUTO,NODISPLAY,NOCMD
- IOS291I NOLOG,NOAUTO,NODISPLAY,NOCMD
- IOS444I NOLOG,NOAUTO,NODISPLAY,NOCMD
- IOS450E NOLOG,NOAUTO,NODISPLAY,NOCMD
- IEA476E NOLOG,NOAUTO,NODISPLAY,NOCMD
- IEA491E NOLOG,NOAUTO,NODISPLAY,NOCMD
- IEA494I NOLOG,NOAUTO,NODISPLAY,NOCMD
- IEA497I NOLOG,NOAUTO,NODISPLAY,NOCMD

Specifying NOLOG, NOAUTO and NODISPLAY together causes the message to be completely deleted.

If your automation has dependencies on any of these messages during a HyperSwap, you should specify AUTO for the message so that it can be seen by your automation.

Exempting messages from processing by Message Flood Automation

If you have automation that assumes that suppression (NODISPLAY) implies no automation (NOAUTO), and the automation is critically dependent on specific messages (for example to signal that a program is UP or DOWN), you may want to code a SPECIFIC message entry for those messages and specify an action of IGNORE which will cause Message Flood Automation to ignore the message even if the specific message threshold is exceeded and other actions have been specified (or defaulted) for the message. Providing a SPECIFIC message entry keeps the message out of both ACTION message and REGULAR message processing.

Since SPECIFIC message processing is limited to a maximum of 30 message specifications, this may not be the appropriate technique to use if you have a large number of messages that you wish to exempt from Message Flood Automation processing. During MPF processing, z/OS will invoke a specific user exit if one is specified in a USEREXIT parameter on the message's MPFLSTxx entry. If the USEREXIT parameter does not exist on the message's MPFLSTxx entry or the message does not have an MPFLSTxx entry, MPF invokes IEAVMXIT, which is where Message Flood Automation runs.

Messages can be easily exempted from Message Flood Automation processing by:

1. creating a very simple MPF exit (that immediately returns to z/OS)
2. creating an MPFLSTxx entry for the message (if it doesn't already have one) and specifying the name of the user exit with the USEREXIT parameter.

The simple MPF exit consists of the following Assembly language code:

```
MPFDMYXT CSECT
MPFDMYXT AMODE 31
MPFDMYXT RMODE ANY
        SR    15,15      Set a normal zero return code.
        BR    14        Return to MPF processing.
        END
```

This code must be assembled using the High-Level Assembler and linked using the z/OS Binder and placed in SYS1.LINKLIB.

The corresponding MPFLSTxx entry would look like the following:

```
message-id,USEREXIT(MPFDMYXT)
```

Example:

```
IEF403I,USEREXIT(MPFDMYXT)
```

You can also specify any of the other MPF parameters such as SUP and AUTO.

You only need to provide a single simple exit; you can specify this same exit for all of the messages that you do not want Message Flood Automation to process.

Using either technique, you of course are taking the risk that these specific messages could cause a message flood, and Message Flood Automation will take no action against them if a flood occurs.

Most modern automation can still obtain messages for automation purposes even if the message is suppressed, so these techniques may not be needed.

Sample PARMLIB specification

The values in the following MSGFLDxx PARMLIB member are for illustration purposes only. You should determine the values that are appropriate for your system.

A sample MSGFLDxx PARMLIB member (named CNZZMFXX) is provided as an example in SYS1.SAMPLIB.

```
/*-----*/
/* Sample MSGFLDxx PARMLIB member. */
/*-----*/
/*-----*/
/* REGULAR message specifications. */
/*-----*/
REGULAR MSGTHRESH=50,JOBTHRESH=20,INTVLTIME=1
```

```

REGULAR SYSIMTIME=2, JOBIMTIME=2
DEFAULT LOG, NOAUTO, NODISPLAY, NOCMD
DEFAULTCMD '&, CANCEL & -- cancelled by Message Flood Automation'
JOB AOC%NV* AUTO
JOB LLA* AUTO
JOB ZAP1 CMD
/*-----*/
/* ACTION message specifications. */
/*-----*/
ACTION MSGTHRESH=50, JOBTHRESH=20, INTVLTIME=1
ACTION SYSIMTIME=2, JOBIMTIME=2
DEFAULT LOG, NOAUTO, NODISPLAY, NOCMD, NORETAIN
DEFAULTCMD '&, CANCEL & -- cancelled by Message Flood Automation'
JOB AOC%NV* AUTO, RETAIN
JOB LLA* AUTO
JOB ZAP2 CMD
/*-----*/
/* SPECIFIC message specifications. */
/*-----*/
SPECIFIC MSGTHRESH=50, INTVLTIME=1
SPECIFIC SYSIMTIME=2
SPECIFIC MSGIMTIME=2
SPECIFIC MSGLIMIT=20
DEFAULT LOG, NOAUTO, NODISPLAY, NORETAIN
MSG IOS001E
MSG IOS003A
MSG IOS050I
MSG IOS051I
MSG IOS071I
MSG IOS251I
MSG IOS444I
MSG IOS450E
*++++*
*++++* end of member *++++*
*++++*

```

ABEND codes

ABEND X'077' RSN X'039'

(CNZZVMXT): The version of the CNZZVMXT message exit and the version of the shared common SQA data area (SQAAREA) are different. CNZZVMXT cannot use the shared common data area because the mapping of the data area may have changed.

This error is usually caused by installing a newer level of the CNZZVMXT message exit without freeing the SQA data area *first* using the SETMF FREE command.

To recover, do the following:

1. Restore the previous version of the CNZZVMXT message exit to the system library (doing an LLA refresh if necessary).
 - o If you restore the previous version of CNZZVMXT by changing the libraries referred to in

the LINKLIST concatenation, you must issue a

- **SETPROG LNKLST,UPDATE,JOB=CONSOLE**

command to cause the CONSOLE address space to use the new LINKLIST concatenation. The K M command runs in the CONSOLE address space.

- *z/OS Initialization and Tuning Reference*: There is some risk associated with the SETPROG command (ABEND106 errors could result) and therefore it is recommended that this command be used only when necessary to prevent an IPL.
2. Activate the message exit using a **K M,UEXIT=Y** command.
 3. Follow the detailed instructions in section "Considerations when migrating from one level to another".

ABEND X'077' RSN X'A24'

(CNZZVMXT): The shared common SQA data area used by Message Flood Automation is not recognizable and cannot be used.

Report this error to IBM Service.

Operator Messages

+---Operator message change summary-----+

NOTE to users of previous levels of Message Flood Automation: All of the MSGF-prefix message IDs used by previous levels of Message Flood Automation have been replaced by CNZZ-prefix message IDs. The numbering and text of the messages has remained the same.

All of the Message Flood Automation module names used by previous levels of Message Flood Automation have likewise been replaced by CNZZ-prefix module names.

+-----+

Message Flood Automation produces the following messages:

CNZZ001I NOT ENOUGH ROOM IN REGULAR JOBS TABLE.

Explanation: The CNZZ001I message can be issued for two reasons.

The REGULAR message job table cannot accommodate another entry. Too many JOB entries were specified in the REGULAR message section of the MSGFLDxx Parmlib member that is being loaded. The REGULAR message job table has a maximum size of 10 entries. Only the first 10 entries will be processed.

The REGULAR message job table cannot accommodate another entry. During REGULAR message intensive mode processing, more address spaces have issued messages than can be tracked in the REGULAR message jobs table. The REGULAR message jobs table has a maximum size of 10 entries. When REGULAR message intensive mode has been entered, only the first 10 address spaces to produce messages are tracked.

System Action: Loading of the MSGFLDxx Parmlib member continues. REGULAR intensive mode processing continues.

Operator Response: None

System Programmer Response: If this message occurs frequently, it is usually an indication that the REGULAR message threshold (MSGTHRESH) has been set too low and needs to be adjusted upward.

Module: CNZZTDP3, CNZZRIMN

Routing Code: 2

Descriptor Code: 5

CNZZ002E MESSAGE THRESHOLD REACHED FOR JOB *jobname*

Explanation: The jobname specified has exceeded the REGULAR job message threshold (JOBTHRESH) and action will be taken against the job. If the jobname matches a JOB entry in the REGULAR message specification, action unique to that JOB entry will be taken. Otherwise, DEFAULT or built-in action will be taken.

In the message text:

jobname

The name of the job that is issuing a large number of messages. NONAME or IEESYSAS indicates that a system service is issuing a large number of messages.

System Action: REGULAR intensive mode processing continues.

Operator Response: Contact the system programmer.

System Programmer Response: This may be an indication that you or the operator should take action against the job since it is exceeding the number of messages specified in your REGULAR job message threshold policy. You should determine whether this is an actual message flooding situation (and perhaps take action if it is) or if your REGULAR job message threshold has perhaps been set too low.

Module: CNZZRACT

Routing Code: 2

Descriptor Code: 3

CNZZ003I NOT ENOUGH ROOM IN MESSAGE TABLE.

Explanation: The SPECIFIC message ID table cannot accommodate another entry. Too many MSG entries were specified in the MSGFLDxx Parmlib member that is being loaded. The SPECIFIC message ID table has a maximum size of 30 entries. Only the first 30 entries will be processed.

System Action: Loading of the MSGFLDxx Parmlib member continues.

Operator Response: None

System Programmer Response: Reduce the number of MSG entries in the MSGFLDxx Parmlib member to no more than 30.

Module: CNZZTDP4

Routing Code: 2

Descriptor Code: 5

CNZZ004E MESSAGES FOR JOB *jobname* NO LONGER ACTED UPON

Explanation: The time between two successive messages from the job exceeds the REGULAR job inter-message time (JOBIMTIME) or the time between two successive messages exceeds the REGULAR system inter-message time (SYSIMTIME) and action will no longer be taken against REGULAR messages from the specified job. Note that this message will not occur if the job ends before its message rate has dropped below the job threshold or the time between two of its messages exceeds the job inter-message time.

In the message text:

jobname

The name of the job that action was being taken against. NONAME or IEESYSAS indicates that action was being taken against a system service.

System Action: Action will no longer be taken against REGULAR messages from the specified job.

Operator Response: Contact the system programmer.

System Programmer Response: If you or the operator are taking action against the job because it was causing a message flooding situation, that action is no longer needed because the job is no longer causing a message flooding situation.

Module: CNZZCKRT, CNZZRIMN

Routing Code: 2

Descriptor Code: 3

CNZZ005E MESSAGES FOR *nnnn* JOBS NO LONGER ACTED UPON. LAST JOB *jobname*

Explanation: The time between two successive REGULAR messages exceeds the system inter-message time (SYSIMTIME) and action will no longer be taken against REGULAR messages from all of the jobs being tracked.

In the message text:

nnnn

The number of jobs for which action will no longer be taken against their messages.

jobname

The name of the last job that action was being taken against. NONAME or IEESYSAS indicates that action was being taken against a system service.

System Action: Action will no longer be taken against REGULAR messages from all of the jobs being tracked.

Operator Response: Contact the system programmer.

System Programmer Response: If you or the operator are taking action against one or more jobs because they were causing a message flooding situation, that action is no longer needed because the jobs are no longer causing a message flooding situation.

Module: CNZZCKRT

Routing Code: 2

Descriptor Code: 3

CNZZ006I Message Flood Automation ABEND *abendcode-reasoncode* at location

Explanation: Message Flood Automation has ABENDED while holding work area storage for intensive mode processing.

In the message text:

abendcode

The ABEND code describing the failure.

reasoncode

The reason code associated with the ABEND code.

location

The name of the load module in which the error occurred followed by the offset of the error within the load module. If the name of the load module cannot be determined, the virtual storage address of the failure is provided instead of the load module name and offset.

System Action: The work area storage is freed and the Message Flood Automation message exit is disabled by the Message Processing Facility. If the error occurred while Parmlib record storage was held, that storage is freed as well.

Operator Response: Contact your system programmer.

System Programmer Response: Report this problem to IBM. Note that if the Message Flood

Automation code is being invoked from your IEAVMXIT routine, your IEAVMXIT code will be disabled by the Message Processing Facility even though the error is in Message Flood Automation. You should be able to turn off Message Flood Automation and restart your IEAVMXIT code by issuing a SETMF OFF command to Message Flood Automation and a K M,UEXIT=Y command to z/OS.

Module: CNZZVMES, CNZZPRTR

Routing Code: 2

Descriptor Code: -

CNZZ007E MESSAGE RATE EXCEEDED *nnnnnn* IN *ssss* SECONDS.

Explanation: The REGULAR message rate threshold has been exceeded and Message Flood Automation is now running in REGULAR message intensive mode to determine what address space is producing the messages.

In the message text:

nnnnnn

The number of REGULAR messages that were counted. This is the REGULAR MSGTHRESH value.

ssss

The number of seconds that it took for the messages to be counted. The time is less than or equal to the REGULAR INTVLTIME value.

System Action: Message Flood Automation begins tracking the address spaces that are producing REGULAR messages. The first 10 address spaces to produce messages will be tracked.

Operator Response: Contact the system programmer.

System Programmer Response: This message should only be produced in a true message flooding situation. If this message occurs frequently, you should review your REGULAR message threshold and interval time specifications and adjust them to achieve a higher threshold. The threshold should be high enough that Message Flood Automation is not constantly oscillating into and out of intensive mode processing.

Module: CNZZCKRT

Routing Code: 2

Descriptor Code: 3

CNZZ008E REGULAR MESSAGE RATE ACCEPTABLE. *nnnnnnnn* MESSAGES ACTED UPON.

Explanation: The message rate has fallen below the REGULAR message threshold and Message Flood

Automation is no longer operating in REGULAR intensive mode.

In the message text:

nnnnnnnn

The number of REGULAR messages that were acted upon during the message flood. If the value is zero, it means that no job exceeded the job threshold while REGULAR message processing was in intensive mode.

System Action: Message Flood Automation terminates REGULAR intensive mode processing and no longer tracks the message production of individual jobs.

Operator Response: Contact the system programmer.

System Programmer Response: This message should only be produced at the end of a true message flooding situation. If this message occurs frequently, you should review your REGULAR message threshold and interval time specifications and adjust them to achieve a higher threshold. The threshold should be high enough that Message Flood Automation is not constantly oscillating into and out of intensive mode processing.

Module: CNZZRIOF

Routing Code: 2

Descriptor Code: 3

CNZZ009E ACTION MSG RATE EXCEEDED *nnnnnn msgid* MSGS IN *ssss* SECS.

Explanation: The ACTION message rate threshold has been exceeded and Message Flood Automation is now running in ACTION message intensive mode to determine what address space is producing the messages.

In the message text:

nnnnnn

The number of ACTION messages that were counted. This is the ACTION MSGTHRESH value.

msgid

The message ID of the ACTION message that exceeded the ACTION message threshold.

ssss

The number of seconds that it took for the messages to be counted. The time is less than or equal to the ACTION INTVLTIME value.

System Action: Message Flood Automation begins tracking the address spaces that are producing ACTION messages. The first 10 address spaces to produce action messages will be tracked.

Operator Response: Contact the system programmer.

System Programmer Response: This message should only be produced in a true message flooding situation. If this message occurs frequently, you should review your ACTION message threshold and interval time specifications and adjust them to achieve a higher threshold. The threshold should be high enough that Message Flood Automation is not constantly oscillating into and out of intensive mode processing.

Module: CNZZCKRT

Routing Code: 2

Descriptor Code: 3

CNZZ010E ACTION MESSAGES FOR JOB *jobname* NO LONGER ACTED UPON

Explanation: The time between two successive messages from the job exceeds the ACTION job inter-message time (JOBIMTIME) or the time between two successive messages exceeds the ACTION system inter-message time (SYSIMTIME) and action will no longer be taken against ACTION messages from the specified job. Note that this message will not occur if the job ends before its message rate has dropped below the job threshold or the time between two of its messages exceeds the job inter-message time.

In the message text:

jobname

The name of the job that action was being taken against. NONAME or IEESYSAS indicates that action was being taken against a system service.

System Action: Action will no longer be taken against ACTION messages from the specified job.

Operator Response: Contact the system programmer.

System Programmer Response: If you or the operator are taking action against the job because it was causing a message flooding situation, that action is no longer needed because the job is no longer causing a message flooding situation.

Module: CNZZCKRT, CNZZAIMN

Routing Code: 2

Descriptor Code: 3

CNZZ012I Message Flood Automation ABEND *abendcode-reasoncode* at *location*

Explanation: Message Flood Automation has ABENDED while holding work area storage for command processing.

In the message text:

abendcode

The ABEND code describing the failure.

reasoncode

The reason code associated with the ABEND code.

location

The name of the load module in which the error occurred followed by the offset of the error within the load module. If the name of the load module cannot be determined, the virtual storage address of the failure is provided instead of the load module name and offset.

System Action: The work area storage is freed and the Message Flood Automation command exit is disabled by the z/OS command exit processing. If the error occurred while holding work area storage for processing the DISPLAY MSGFLD,MSGRATE command, that storage is freed as well.

Operator Response: Contact your system programmer.

System Programmer Response: Report this problem to IBM.

Module: CNZZCMES

Routing Code: 2

Descriptor Code: -

CNZZ013I Operation cannot be performed. status

Explanation: The version of the CNZZCMXT command exit code and the version of the shared common SQA data area (SQAAREA) are different. CNZZCMXT cannot use the shared common data area because the mapping of the data area may have changed.

In the message text:

status

One of the following:

CNZZCMXT/SQAAREA mismatch.

CNZZCMXT is unable to determine the version of the shared common data area (SQAAREA) and cannot safely use it. The shared common data area either lacks version information or the version information has been corrupted.

CNZZCMXT > SQAAREA/CNZZVMXT

CNZZCMXT has determined that it is more current than the shared common data area (SQAAREA) created by the CNZZVMXT message exit and cannot safely use it.

CNZZCMXT < SQAAREA/CNZZVMXT

CNZZCMXT has determined that it is not as current as the shared common data area (SQAAREA) created by the CNZZVMXT message exit and cannot safely use it.

System Action: The requested operation is not performed.

Operator Response: Report this problem to the System Programmer.

System Programmer Response: See the detailed recovery scenarios provided in the Message Flood Automation User's Guide.

Module: CNZZCMXT

Routing Code:

* The message will be routed back to the console that initiated the associated request.

Descriptor Code: 5

CNZZ014E ACTION MESSAGE THRESHOLD REACHED FOR JOB *jobname*

Explanation: The jobname specified has exceeded the ACTION job message threshold (JOBTHRESH) and action will be taken against the job. If the jobname matches a JOB entry in the ACTION message specification, action unique to that JOB entry will be taken. Otherwise, DEFAULT or built-in action will be taken.

In the message text:

jobname

The name of the job that is issuing a large number of action messages. NONAME or IEESYSAS indicates that a system service is issuing a large number of messages.

System Action: ACTION intensive mode processing continues.

Operator Response: Contact the system programmer.

System Programmer Response: This may be an indication that you or the operator should take action against the job since it is exceeding the number of messages specified in your ACTION job message threshold policy. You should determine whether this is an actual message flooding situation (and perhaps take action if it is) or if your ACTION job message threshold has perhaps been set too low.

Module: CNZZAACT

Routing Code: 2

Descriptor Code: 3

CNZZ015E ACTION MSGS FOR *nnnn* JOBS NO LONGER ACTED UPON. LAST JOB

jobname

Explanation: The time between two successive ACTION messages exceeds the ACTION system inter-message time (SYSIMTIME) and action will no longer be taken against ACTION messages from all of the jobs that were being tracked.

In the message text:

nnnn

The number of jobs that will no longer have their messages acted upon.

jobname

The name of the last job that action was being taken against. NONAME or IEESYSAS indicates that action was being taken against a system service.

System Action: Action will no longer be taken against ACTION messages from the affected jobs.

Operator Response: Contact the system programmer.

System Programmer Response: If you or the operator are taking action against one or more jobs because they were causing a message flooding situation, that action is no longer needed because the jobs are no longer causing a message flooding situation.

Module: CNZZCKRT

Routing Code: 2

Descriptor Code: 3

CNZZ016I Message Flood Automation policy initialized.

Explanation: Message CNZZ016I is issued in response to the SET MSGFLD=xx command. The message indicates that the requested Parmlib member was read and that Message Flood Automation parameters were successfully re-initialized.

System Action: If Message Flood Automation has been enabled, Message Flood Automation uses the new parameters.

Operator Response: If Message Flood Automation was not previously enabled, and you wish to use the new parameters, you should issue a SETMF ON command to enable Message Flood Automation processing.

System Programmer Response: None.

Module: CNZZINIT

Routing Code: 2

Descriptor Code: 5

CNZZ017I Previous PARMLIB read already underway; try later.

Explanation: Message CNZZ017I is issued in response to the SET MSGFLD=xx command. The message indicates that a previous SET MSGFLD=xx command is still being processed and that another SET MSGFLD=xx command cannot be processed until the previous command completes.

System Action: Processing of the previous SET MSGFLD=xx command continues. Processing of this SET MSGFLD=xx command is terminated.

Operator Response: Wait a brief period of time and re-enter the SET MSGFLD=xx command. If this message continues to reappear, contact the system programmer.

System Programmer Response: If this message occurs again after waiting several minutes, contact IBM.

Module: CNZZCMXT

Routing Code: -

Descriptor Code: -

CNZZ018E ACTION MESSAGE RATE ACCEPTABLE. *nnnnnnnn* MESSAGES ACTED UPON.

Explanation: The message rate has fallen below the ACTION message threshold and Message Flood Automation is no longer operating in ACTION intensive mode.

In the message text:

nnnnnnnn

The number of ACTION messages that were acted upon during the message flood. If the value is zero, it means that no job exceeded the job threshold while ACTION message processing was in intensive mode.

System Action: Message Flood Automation terminates ACTION intensive mode processing and no longer tracks the message production of individual jobs.

Operator Response: Contact the system programmer.

System Programmer Response: This message should only be produced at the end of a true message flooding situation. If this message occurs frequently, you should review your ACTION message threshold and interval time specifications and adjust them to achieve a higher threshold. The threshold should be high enough that Message Flood Automation is not constantly oscillating into and out of intensive mode processing.

Module: CNZZAIOF

Routing Code: 2

Descriptor Code: 3

CNZZ019I NOT ENOUGH ROOM IN ACTION JOBS TABLE.

Explanation: The CNZZ019I message can be issued for two reasons.

The ACTION message job table cannot accommodate another entry. Too many JOB entries were specified in the ACTION message section of the MSGFLDxx Parmlib member that is being loaded. The ACTION message job table has a maximum size of 10 entries. Only the first 10 entries will be processed.

The ACTION message job table cannot accommodate another entry. During ACTION message intensive mode processing, more address spaces have issued messages than can be tracked in the ACTION message jobs table. The ACTION message jobs table has a maximum size of 10 entries. When ACTION message intensive mode has been entered, only the first 10 address spaces to produce action messages are tracked.

System Action: Loading of the MSGFLDxx Parmlib member continues. ACTION intensive mode processing continues.

Operator Response: None

System Programmer Response: If this message occurs frequently, it is usually an indication that the ACTION message threshold (MSGTHRESH) has been set too low and needs to be adjusted upward. The threshold should be high enough that Message Flood Automation is not constantly oscillating into and out of intensive mode processing.

Module: CNZZTDP3, CNZZAIMN

Routing Code: 2

Descriptor Code: 5

CNZZ022I Operation cannot be performed. No Message Flood SQA area.

Explanation: The requested operation requires manipulating information held in the Message Flood Automation shared common storage area (SQA) but no SQA area could be located by Message Flood Automation command processing. This may indicate that: (1) The IEAVMXIT (CNZZVMXT) message exit was not loaded or activated. (2) The name/token anchor of the shared common storage area (SQA) could not be located. (3) The shared common storage area was explicitly freed using a SETMF FREE command.

System Action: The requested operation is terminated.

Operator Response: To recover from this situation, issue the K M,UEXIT=Y command to re-instate the IEAVMXIT (CNZZVMXT) exit. If this message reoccurs after issuing the K M,UEXIT=Y command, report the problem to the system programmer.

System Programmer Response: If issuing the K M,UEXIT=Y command does not resolve the problem, it is likely that the IEAVMXIT (CNZZVMXT) message exit was not installed or was installed incorrectly. The operating system attempts to load IEAVMXIT early in IPL processing, so examine SYSLOG for messages indicating that IEAVMXIT was not successfully loaded.

Module: CNZZCMXT

Routing Code:

* The message will be routed back to the console that initiated the associated request.

Descriptor Code: 5

CNZZ026I Message Flood Automation SQA area FREEMAINED.

Explanation: Message CNZZ026I is issued in response to the SETMF FREE command. The shared common storage area (SQA) has been FREEMAINED and Message Flood Automation message processing has been placed into the INHIBITED state. The INHIBITED state prevents the shared common storage area (SQA) from being automatically re-acquired. This allows Message Flood Automation to be completely shutdown without any storage remaining.

System Action: Message Flood Automation continues in the INHIBITED state.

Operator Response: If you intend to shutdown Message Flood Automation, you can now issue the K M,UEXIT=N command and have Message Flood Automation free all of its remaining storage and terminate. You can remove the INHIBITED state and re-instate the Message Flood Automation IEAVMXIT (CNZZVMXT) message exit by issuing a K M,UEXIT=Y command.

System Programmer Response: None.

Module: CNZZCMXT

Routing Code:

* The message will be routed back to the console that initiated the associated request.

Descriptor Code: 5

CNZZ027I Cannot free SQA while Message Flood Automation ENABLED.

Explanation: Message CNZZ027I is issued in response to the SETMF FREE command. The command cannot be processed unless Message Flood Automation is in the DISABLED state. Message Flood Automation can be placed in the DISABLED state using the SETMF OFF command.

System Action: Message Flood Automation continues in the ENABLED state.

Operator Response: If you intend to free the Message Flood Automation shared common storage area (SQA), issue the SETMF OFF command to place Message Flood Automation into the DISABLED state and then re-issue the SETMF FREE command to free the common storage area.

System Programmer Response: None.

Module: CNZZCMXT

Routing Code:

* The message will be routed back to the console that initiated the associated request.

Descriptor Code: 5

CNZZ028I No Message Flood SQA to FREEMAIN.

Explanation: Message CNZZ028I is issued in response to the SETMF FREE command. The shared common storage area (SQA) has already been freed or was never acquired.

System Action: Message Flood Automation remains in its current state.

Operator Response: If you intend to shutdown Message Flood Automation, you can issue the K M,UEXIT=N command and have Message Flood Automation free any of its remaining storage and terminate. You can re-instate the Message Flood Automation IEAVMXIT (CNZZVMXT) message exit by issuing a K M,UEXIT=Y command.

System Programmer Response: None.

Module: CNZZCMXT

Routing Code: 2

Descriptor Code: -

CNZZ029I Cannot free SQA while Message Rate Monitoring active.

Explanation: Message CNZZ029I is issued in response to the SETMF FREE command. The command cannot be processed while Message Rate Monitoring is active because Message Rate Monitoring uses the shared common storage area to record its statistics.

System Action: Message Flood Automation remains in its current state.

Operator Response: If you wish to free the Message Flood Automation shared common storage area, you must first stop Message Rate Monitoring by issuing a SETMF MONITOROFF command and then re-issue the SETMF FREE command.

System Programmer Response: None.

Module: CNZZCMXT

Routing Code:

* The message will be routed back to the console that initiated the associated request.

Descriptor Code: 5

CNZZ031E SPECIFIC MSG RATE EXCEEDED *nnnnnn* MSGS IN *ssss* SECS.

Explanation: The SPECIFIC message rate threshold has been exceeded and Message Flood Automation is now running in SPECIFIC message intensive mode to determine what message ID is responsible for producing the messages.

In the message text:

nnnnnn

The number of SPECIFIC messages that were counted. This is the SPECIFIC MSGTHRESH value.

ssss

The number of seconds that it took for the messages to be counted. The time is less than or equal to the SPECIFIC INTVLTIME value.

System Action: Message Flood Automation begins tracking the message IDs that are producing SPECIFIC messages.

Operator Response: Contact the system programmer.

System Programmer Response: This message should only be produced in a true message flooding situation. If this message occurs frequently, you should review your SPECIFIC message threshold and interval time specifications and adjust them to achieve a higher threshold. The threshold should be high enough that Message Flood Automation is not constantly oscillating into and out of intensive mode processing.

Module: CNZZCKRT

Routing Code: 2

Descriptor Code: 3

CNZZ032E SPECIFIC MESSAGE RATE ACCEPTABLE. *nnnnnnnn* MESSAGES ACTED UPON.

Explanation: The message rate has fallen below the SPECIFIC message threshold and Message Flood Automation is no longer operating in SPECIFIC intensive mode.

In the message text:

nnnnnnnn

The number of SPECIFIC messages that were acted upon during the message flood. If the value is zero, it means that no message ID exceeded the message threshold while SPECIFIC message processing was in intensive mode.

System Action: Message Flood Automation terminates SPECIFIC intensive mode processing and no longer tracks the message production of individual message IDs.

Operator Response: Contact the system programmer.

System Programmer Response: This message should only be produced at the end of a true message flooding situation. If this message occurs frequently, you should review your SPECIFIC message threshold and interval time specifications and adjust them to achieve a higher threshold. The threshold should be high enough that Message Flood Automation is not constantly oscillating into and out of intensive mode processing.

Module: CNZZSIOF

Routing Code: 2

Descriptor Code: 3

CNZZ033E SPECIFIC MESSAGE THRESHOLD REACHED FOR *msgid*

Explanation: The *msgid* specified has exceeded the SPECIFIC message threshold (MSGTHRESH) and action will be taken against the *msgid*. The *msgid* matches a MSG entry in the SPECIFIC message specification and action unique to that MSG entry will be taken if actions were defined for the *msgid*. Otherwise, built-in or DEFAULT actions will be taken.

In the message text:

msgid

The message ID of the SPECIFIC message that exceeded the SPECIFIC message threshold.

System Action: SPECIFIC intensive mode processing continues.

Operator Response: Contact the system programmer.

System Programmer Response: This may be an indication that you or the operator should take action since this message ID is exceeding the number of messages specified in your SPECIFIC message threshold policy. You should determine whether this is an actual message flooding situation (and perhaps take action if it is) or if your SPECIFIC message threshold has perhaps been set too low.

Module: CNZZSIMN

Routing Code: 2

Descriptor Code: 3

CNZZ034E SPECIFIC MESSAGE *msgid* NO LONGER ACTED UPON

Explanation: The time between two successive messages exceeds the SPECIFIC message inter-message time (MSGIMTIME) or the time between two successive messages exceeds the SPECIFIC system inter-message time (SYSIMTIME) and action will no longer be taken against this *msgid*. Note

that this message will not occur if the source of the messages ends before its message rate has dropped below the message threshold or the time between two of its messages exceeds the message inter-message time.

In the message text:

msgid

The message ID of the SPECIFIC message that will no longer be acted upon.

System Action: Action will no longer be taken against the specific msgid.

Operator Response: Contact the system programmer.

System Programmer Response: If you or the operator are taking action because this message ID was causing a message flooding situation, that action is no longer needed because this message ID is no longer causing a message flooding situation.

Module: CNZZCKRT, CNZZSIMN

Routing Code: 2

Descriptor Code: 3

CNZZ035E *nnnn* SPECIFIC MESSAGES NO LONGER ACTED UPON.

Explanation: The time between two successive messages exceeds the SPECIFIC message inter-message time (MSGIMTIME) or the time between two successive messages exceeds the SPECIFIC system inter-message time (SYSIMTIME) and action will no longer be taken against multiple message-ids.

In the message text:

nnnn

The number of message-ids that will no longer be acted upon.

System Action: Action will no longer be taken against any specific message-ids.

Operator Response: Contact the system programmer.

System Programmer Response: If you or the operator are taking action because multiple message IDs were causing a message flooding situation, that action is no longer needed because these message IDs are no longer causing a message flooding situation.

Module: CNZZCKRT

Routing Code: 2

Descriptor Code: 3

CNZZ040I Intensive modes: REGULAR-*st1* ACTION-*st2* SPECIFIC-*st3*

Explanation: Message CNZZ040I is issued in response to the DISPLAY MSGFLD,MODE command and indicates the current state of intensive mode processing. All of the intensive mode states should be OFF unless a message flooding situation is underway.

In the message text:

st1

The state of REGULAR intensive mode: either ON or OFF.

st2

The state of ACTION intensive mode: either ON or OFF.

st3

The state of SPECIFIC intensive mode: either ON or OFF.

System Action: None

Operator Response: None

System Programmer Response: None

Module: CNZZCMXT

Routing Code:

* The message will be routed back to the console that initiated the associated request.

Descriptor Code: 5

CNZZ041I Message Flood Automation *state* PARMLIB member: *member*

Explanation: Message CNZZ041I is issued in response to the SETMF ON and SETMF OFF commands and indicates the state of Message Flood Automation after the requested operation has been performed.

In the message text:

state

The state of Message Flood Automation: either ENABLED, DISABLED or INHIBITED. In the ENABLED state, Message Flood Automation will take action if a message flood occurs. In the DISABLED state, Message Flood Automation will take no action if a message flood occurs. The INHIBITED state indicates that a SETMF FREE command has freed the Message Flood Automation shared common data area.

member

The name of the currently loaded MSGFLDxx Parmlib member. If the name is "internal", no MSGFLDxx Parmlib member has been loaded and Message Flood Automation will use its internal defaults if it is ENABLED.

System Action: If the state is ENABLED, Message Flood Automation will use the currently active policy, as acquired from the MSGFLDxx Parmlib member, or its own built-in defaults, to determine when a message flooding situation is underway, and take action should a message flooding situation occur.

If the state is DISABLED, Message Flood Automation does not look at the message traffic and will take no action should a message flooding situation occur.

If the state is INHIBITED, Message Flood Automation does not look at the message traffic and will take no action should a message flooding situation occur.

Operator Response: If the state is INHIBITED, and you wish to remove the INHIBITED state, you must deactivate and reactivate the IEAVMXIT message exit using the K M,UEXIT= command.

System Programmer Response: None

Module: CNZZCMXT

Routing Code:

* The message will be routed back to the console that initiated the associated request.

Descriptor Code: 5

CNZZ042I Message Flood Automation *vvvvvvv xxxxxxxxx*
Policy *pppppppppppppp* **Using PARMLIB member:** *membernm*
Intensive modes:REGULAR-*rst* ACTION-*ast* SPECIFIC-*sst*
Message rate monitoring *mrmstate. mmmmmmmmm* **msgs** *sssssssss* **secs**

Explanation: Message CNZZ042I is issued in response to the DISPLAY MSGFLD,STATUS command and provides the current status of a number of Message Flood Automation functions.

In the message text:

vvvvvvv

The Message Flood Automation version, release and modification level in the form VnRnMnn.

xxxxxxxxx

The state of Message Flood Automation: either ENABLED, DISABLED or INHIBITED. In the ENABLED state, Message Flood Automation will take action if a message flood occurs. In the DISABLED state, Message Flood Automation will take no action if a message flood occurs. The INHIBITED state indicates that a SETMF FREE command has freed the Message Flood

Automation shared common data area.

PPPPPPPPPPPPPP

The state of Message Flood Automation policy: either UNINITIALIZED or INITIALIZED.

membernm

The name of the currently loaded MSGFLDxx Parmlib member. If the name is "internal", no MSGFLDxx Parmlib member has been loaded and Message Flood Automation will use its internal defaults if it is ENABLED.

rst

The state of REGULAR message intensive mode: either OFF or ON.

ast

The state of ACTION message intensive mode: either OFF or ON.

sst

The state of SPECIFIC message intensive mode: either OFF or ON.

mrnstate

The state of message rate monitoring: either DISABLED or ENABLED

mmmmmmmmmm

The number of messages that have been counted since Message Rate Monitoring was enabled.

ssssssss

The number of seconds that have elapsed since Message Rate Monitoring was enabled.

System Action: If the state is ENABLED, Message Flood Automation will use the currently active policy, as acquired from the MSGFLDxx Parmlib member, or its own built-in defaults, to determine when a message flooding situation is underway, and take action should a message flooding situation occur.

If the state is DISABLED, Message Flood Automation does not look at the message traffic and will take no action should a message flooding situation occur.

If the state is INHIBITED, Message Flood Automation does not look at the message traffic and will take no action should a message flooding situation occur.

Operator Response: If the state is INHIBITED, and you wish to remove the INHIBITED state, you must deactivate and reactivate the IEAVMXIT message exit using the K M,UEXIT= command.

System Programmer Response: None

Module: CNZZSTAT

Routing Code:

* The message will be routed back to the console that initiated the associated request.

Descriptor Code: 5

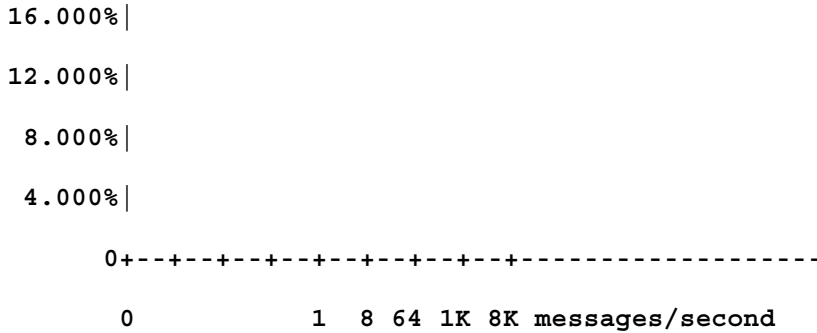
CNZZ043I Message Flood Automation

Instantaneous Message Rates

nnnnnnnnn messages in *ssssssss* seconds *mmmmmmmm* msg/sec

% of time at msg rate *cccccccc* messages w/most common rate

100.000% |
96.000% |
92.000% |
88.000% |
84.000% |
80.000% |
76.000% |
72.000% |
68.000% |
64.000% |
60.000% |
56.000% |
52.000% |
48.000% |
44.000% |
40.000% |
36.000% |
32.000% |
28.000% |
24.000% |
20.000% |



Suggested threshold for $p1\%$ is $t1$
Suggested threshold for $p2\%$ is $t2$
Suggested threshold for $p3\%$ is $t3$
Suggested threshold for $p4\%$ is $t4$
Suggested threshold for $p5\%$ is $t5$

Explanation: Message CNZZ043I is issued in response to the DISPLAY MSGFLD,MSGRATE command and provides information about the message rates observed by the Message Rate Monitoring function. The graph shows the percent of time at a message rate on the Y-axis and instantaneous message rates (in messages / second) on the X-axis. The X-axis scale is logarithmic with each character position being a factor of 2 greater than the previous position in the rightward direction. Tickmarks are provided at 8X intervals. The Y-axis is of variable length, with a default length of 25 lines. Any value from 0 to 999 is acceptable on the command, but it will be adjusted to the range 10 to 200. The actual value used is selected to provide readable increments on the Y-axis.

Each vertical bar of asterisks in the graph is rightward cumulative, that is each bar represents not only the fraction of time at its own rate, but the fraction of time with a lesser rate. (A bar's own contribution to the time at a given message rate is therefore the difference between its height and the height of its immediate leftward neighbor).

A vertical line (|) indicates the most common message rate. On the X-axis, the minimum and maximum message rates recorded are indicated (by the > and < symbols, respectively) on either side of the mean message rate. The percentage of messages occurring at the maximum message rate is usually quite small and may not be visible unless the resolution of the graph is improved by increasing the number of lines in the graph.

The graph should have a characteristic "S" shape to it caused by there being relatively few messages occurring at very low message rates (the bottom left of the "S" curve) and very few messages occurring at very high message rates (the top right of the "S" curve).

The graph presents instantaneous message rates that are determined from the inter-arrival times of the messages. Small inter-arrival times result in high instantaneous message rates. Large inter-arrival times result in low instantaneous message rates. A high message rate on the graph does not necessarily imply that multiple, consecutive messages were issued at that rate. It is quite possible for a high message rate to be indicated without Message Flood Automation being triggered. (It is multiple, consecutive, high message rate messages that trigger Message Flood Automation). Message Flood Automation thresholds

should be set based on the mean (most common) message rate, not on the maximum message rates.

In the message text:

nnnnnnnnnn

The number of messages that have been counted since Message Rate Monitoring was enabled.

sssssssss

The number of seconds that have elapsed since Message Rate Monitoring was enabled.

mmmmmmmmmm

The average message rate, in messages / second.

cccccccc

The number of messages with the most commonly occurring message rate. In the context used here, this interpreted as the most commonly occurring inter-arrival time -- the time between two successive messages.

p1

The percent of time that the message rate does not exceed the suggested threshold.

t1

The suggested threshold, in messages per second.

p2

The percent of time that the message rate does not exceed the suggested threshold.

t2

The suggested threshold, in messages per second.

p3

The percent of time that the message rate does not exceed the suggested threshold.

t3

The suggested threshold, in messages per second.

p4

The percent of time that the message rate does not exceed the suggested threshold.

t4

The suggested threshold, in messages per second.

p5

The percent of time that the message rate does not exceed the suggested threshold.

t5

The suggested threshold, in messages per second.

System Action: None

Operator Response: None

System Programmer Response: The suggested threshold values can be used to set the REGULAR message MSGTHRESH value.

Module: CNZZGRAF

Routing Code:

* The message will be routed back to the console that initiated the associated request.

Descriptor Code: 5

CNZZ044I No message rate data to display

Explanation: Message CNZZ044I is issued in response to the DISPLAY MSGFLD,MSGRATE command if there is no message rate data to be displayed. Message rate data collection is enabled using the SETMF MONITORON command. It is also possible to receive this message if no messages were monitored between the time that the MONITORON command was issued and the DISPLAY MSGFLD,MSGRATE command was issued.

System Action: None

Operator Response: If you wish to collect Message Rate Monitoring data, issue a SETMF MONITORON command to enable Message Rate Monitoring if you have not already done so.

System Programmer Response: None.

Module: CNZZGRAF

Routing Code:

* The message will be routed back to the console that initiated the associated request.

Descriptor Code: 5

CNZZ045I Graph length too short; set to minimum length

Explanation: Message CNZZ045I is issued in response to the DISPLAY MSGFLD,MSGRATE command and indicates that the supplied graph length is less than 8 lines. The length of the graph is reset to the minimum length of 8 lines.

System Action: A graph 8 lines in length is produced.

Operator Response: Supply a graph length of at least 8 lines.

System Programmer Response: None.

Module: CNZZGRAF

Routing Code:

* The message will be routed back to the console that initiated the associated request.

Descriptor Code: 5

CNZZ046I Graph length too long; set to maximum length

Explanation: Message CNZZ046I is issued in response to the DISPLAY MSGFLD,MSGRATE command and indicates that the supplied graph length is greater than 200 lines. The length of the graph is reset to the maximum length of 200 lines.

System Action: A graph 200 lines in length is produced.

Operator Response: Supply a graph length of 200 or fewer lines.

System Programmer Response: None.

Module: CNZZGRAF

Routing Code:

* The message will be routed back to the console that initiated the associated request.

Descriptor Code: 5

CNZZ101I Message flood name/token anchor *status*

Explanation: Message Flood Automation has attempted to create or delete the name/token that is used to anchor the shared common storage area (SQA).

In the message text:

status

One of the following:

deleted.

The Message Flood Automation name/token anchor was successfully deleted as part of a K M,UEXIT=N or K M,UEXIT=Y operation.

created.

The Message Flood Automation name/token anchor was successfully created as part of a K M,UEXIT=Y operation.

not deleted.

The Message Flood Automation name/token anchor was not successfully deleted as part of a K M,UEXIT=N or K M,UEXIT=Y operation.

not created.

The Message Flood Automation name/token anchor was not successfully created as part of a K M,UEXIT=Y operation.

System Action: Message Flood Automation processing continues.

Operator Response: Contact your system programmer.

System Programmer Response: If a return code is produced, report this to IBM Service.

Module: CNZZVMXT

Routing Code: 2

Descriptor Code: -

CNZZ102I Message Flood Automation anchor *status*

Explanation: Message Flood Automation has determined that the name/token used to anchor the shared common storage area (SQA) already exists and does not need to be re-created or does not exist and needs to be re-created.

In the message text:

status

One of the following:

already exists.

The Message Flood Automation name/token anchor already exists and Message Flood

Automation was able to successfully re-attach to its data structures.

does not exist.

The Message Flood Automation name/token anchor does not exist and Message Flood Automation was unable to successfully re-attach to its data structures.

System Action: Message Flood Automation processing continues.

Operator Response: Contact the system programmer.

System Programmer Response: If the "does not exist" form of the message is produced, contact IBM Service.

Module: CNZZVMXT, CNZZSCHD

Routing Code: 2

Descriptor Code: -

CNZZ103I Message Flood Automation command processing anchored.

Explanation: Message Flood Automation command processing has been able to re-establish access to the shared common storage area (SQA) that it shares with Message Flood Automation message processing.

System Action: Message Flood Automation processing continues.

Operator Response: None.

System Programmer Response: None.

Module: CNZZCMXT

Routing Code:

* The message will be routed back to the console that initiated the associated request.

Descriptor Code: 5

CNZZ104I Message Flood Automation command processing terminated.

Explanation: Message Flood Automation command processing has been terminated by the system, typically in response to a SET MPF= command.

System Action: Message Flood Automation is no longer able to react to Message Flood Automation commands.

Operator Response: If you wish to continue to be able to issue Message Flood Automation commands,

you must reinstate the Message Flood Automation command exit by issuing a SET MPF= command for an MPFLSTxx member that contains a .CMD statement for the CNZZCMXT module.

System Programmer Response: None.

Module: CNZZCMXT

Routing Code: 2

Descriptor Code: -

CNZZ202I No keywords specified. *errtxt*

Explanation: The CNZZ202I message can be produced by either MSGFLDxx Parmlib member processing or by command processing. During the processing of a REGULAR, ACTION, SPECIFIC or DEFAULT MSGFLDxx Parmlib statement, a keyword was expected but none was provided. During the processing of a DISPLAY MSGFLD,MSGTYPE= msgtype,keyword or SETMF MSGTYPE=msgtype,keyword=value command, a keyword was expected but none was provided.

In the message text:

errtxt

The first 25 characters of the string that is in error.

System Action: The statement or command is not processed.

Operator Response: Correct and re-issue the DISPLAY MSGFLD or SETMF MSGTYPE command.

System Programmer Response: Correct the appropriate MSGFLDxx Parmlib statement and re-load the MSGFLDxx Parmlib member.

Module: CNZZCMXT, CNZZTDP2

Routing Code: 2

Descriptor Code: 5

CNZZ203I Syntax error. No = in string *errtxt*

Explanation: The CNZZ203I message can be produced by either MSGFLDxx Parmlib member processing or by command processing. During the processing of a REGULAR, ACTION or SPECIFIC MSGFLDxx Parmlib statement, an equal sign character was expected but was not provided. During the processing of a DISPLAY MSGFLD,MSGTYPE= msgtype,keyword or SETMF MSGTYPE=msgtype,keyword=value command, an equal sign character was expected but was not provided.

In the message text:

errtxt

The first 25 characters of the string that is in error.

System Action: The statement or command is not processed.

Operator Response: Correct and re-issue the DISPLAY MSGFLD or SETMF MSGTYPE command.

System Programmer Response: Correct the appropriate MSGFLDxx Parmlib statement and re-load the MSGFLDxx Parmlib member.

Module: CNZZTOKN

Routing Code: 2

Descriptor Code: 5

CNZZ204I Syntax error. No keyword in string *errtxt*

Explanation: The CNZZ204I message can be produced by either MSGFLDxx Parmlib member processing or by command processing. During the processing of a REGULAR, ACTION or SPECIFIC MSGFLDxx Parmlib statement, a keyword was expected but was not provided. During the processing of a DISPLAY MSGFLD,MSGTYPE= msgtype,keyword or SETMF MSGTYPE=msgtype,keyword=value command, a keyword was expected but was not provided.

In the message text:

errtxt

The first 25 characters of the string that is in error.

System Action: The statement or command is not processed.

Operator Response: Correct and re-issue the DISPLAY MSGFLD or SETMF MSGTYPE command.

System Programmer Response: Correct the appropriate MSGFLDxx Parmlib statement and re-load the MSGFLDxx Parmlib member.

Module: CNZZTOKN

Routing Code: 2

Descriptor Code: 5

CNZZ205I Syntax error. No value in string *errtxt*

Explanation: The CNZZ205I message can be produced by either MSGFLDxx Parmlib member processing or by command processing. During the processing of a REGULAR, ACTION or SPECIFIC MSGFLDxx Parmlib statement, a value was expected but was not provided. During the processing of a DISPLAY MSGFLD,MSGTYPE= msgtype,keyword or SETMF MSGTYPE=msgtype,keyword=value command, a value was expected but was not provided.

In the message text:

errtxt

The first 25 characters of the string that is in error.

System Action: The statement or command is not processed.

Operator Response: Correct and re-issue the DISPLAY MSGFLD or SETMF MSGTYPE command.

System Programmer Response: Correct the appropriate MSGFLDxx Parmlib statement and re-load the MSGFLDxx Parmlib member.

Module: CNZZTOKN

Routing Code: 2

Descriptor Code: 5

CNZZ206I Non-numeric value in string *errtxt*

Explanation: The CNZZ206I message can be produced by either MSGFLDxx Parmlib member processing or by command processing. During the processing of a REGULAR, ACTION or SPECIFIC MSGFLDxx Parmlib statement, a numeric value was expected but a non-numeric value was provided. During the processing of a DISPLAY MSGFLD,MSGTYPE= msgtype,keyword or SETMF MSGTYPE=msgtype,keyword=value command, a numeric value was expected but a non-numeric value was provided.

In the message text:

errtxt

The first 25 characters of the string that is in error.

System Action: The statement or command is not processed.

Operator Response: Correct and re-issue the DISPLAY MSGFLD or SETMF MSGTYPE command.

System Programmer Response: Correct the appropriate MSGFLDxx Parmlib statement and re-load the MSGFLDxx Parmlib member.

Module: CNZZTOKN

Routing Code: 2

Descriptor Code: 5

CNZZ207I Invalid keyword *errtxt*

Explanation: The CNZZ207I message can be produced by either MSGFLDxx Parmlib member processing or by command processing. During the processing of a REGULAR, ACTION, SPECIFIC, DEFAULT, JOB or MSG MSGFLDxx Parmlib statement, a keyword was provided that is not a valid keyword. During the processing of a DISPLAY MSGFLD,MSGTYPE= msgtype,keyword or SETMF MSGTYPE=msgtype,keyword=value command, a keyword was provided that is not a valid keyword.

In the message text:

errtxt

The first 25 characters of the string that is in error. The keyword may have been misspelled or may not be valid for the particular MSGFLDxx Parmlib statement or command.

System Action: The statement or command is not processed.

Operator Response: Correct and re-issue the DISPLAY MSGFLD or SETMF MSGTYPE command.

System Programmer Response: Correct the appropriate MSGFLDxx Parmlib statement and re-load the MSGFLDxx Parmlib member.

Module: CNZZTOKN, CNZZTDP2, CNZZTDP3, CNZZTDP4, CNZZDVL1, CNZZDVL2, CNZZDVL3, CNZZDVL4, CNZZCMXT

Routing Code: 2

Descriptor Code: 5

CNZZ208I Requested value(s) updated

Explanation: Message CNZZ208I is issued in response to the SETMF MSGTYPE=msgtype,keyword=value command. The requested value(s) have been updated.

System Action: Message Flood Automation will use the updated values.

Operator Response: None.

System Programmer Response: None.

Module: CNZZCMXT

Routing Code:

* The message will be routed back to the console that initiated the associated request.

Descriptor Code: 5

CNZZ209I Action invalid for SPECIFIC msgtype

Explanation: The requested action is not valid for the SPECIFIC message type.

System Action: The requested action is not processed.

Operator Response: Correct and re-enter the operator command, specifying only actions that are valid for the SPECIFIC msgtype.

System Programmer Response: Correct the actions specified in the MSGFLDxx Parmlib member and reload it.

Module: CNZZTDP2, CNZZTDP3, CNZZTDP4

Routing Code: 2

Descriptor Code: -

CNZZ210I Value must be non-zero *nnnnnnnnnn*

Explanation: The specified value must be non-zero.

In the message text:

nnnnnnnnnn

The value that is in error.

System Action: The specified value is not processed.

Operator Response: Correct the value specified and re-enter the command.

System Programmer Response: Correct the value specified in the MSGFLDxx Parmlib member and reload it.

Module: CNZZTOKN

Routing Code: 2

Descriptor Code: 5

CNZZ211I Syntax error. *errtxt*

Explanation: The string shown contains a syntax error that prevents the Message Flood Automation Parmlib statement or operator command from being properly processed.

In the message text:

errtxt

The first 25 characters of the string that is in error. A keyword may have been misspelled or may not be valid for the particular Parmlib statement or operator command.

System Action: The statement or command is not processed.

Operator Response: Correct and re-issue the DISPLAY MSGFLD or SETMF MSGTYPE command.

System Programmer Response: Correct the appropriate MSGFLDxx Parmlib statement and re-load the MSGFLDxx Parmlib member.

Module: CNZZTDP1, CNZZTDP2, CNZZTDP3, CNZZTDP4

Routing Code: 2

Descriptor Code: 5

CNZZ212I Value(s) in error NOT updated.

Explanation: Message CNZZ212I is issued in response to the SETMF MSGTYPE=msgtype,keyword=value command. Due to errors indicated in previous error messages, the requested value(s) have NOT been updated.

System Action: The requested command is not processed.

Operator Response: Fix the error(s) indicated in the previous error message(s) and re-enter the command.

System Programmer Response: None.

Module: CNZZCMXT

Routing Code:

* The message will be routed back to the console that initiated the associated request.

Descriptor Code: 5

CNZZ213I Syntax error: jobname missing

Explanation: Message CNZZ213I is issued during the processing of both JOB statements and SETMF commands containing a JOB= parameter. In the case of a JOB statement, the jobname is either not present or is separated from the JOB specification by more than one blank. In the case of a JOB= parameter, the jobname is either not present or is separated from the JOB= specification by a blank.

System Action: The JOB Parmlib statement or SETMF command is not processed.

Operator Response: Correct the JOB= specification and re-enter the SETMF command.

System Programmer Response: Correct the JOB Parmlib statement and re-load the MSGFLDxx Parmlib member.

Module: CNZZTDP3

Routing Code: 2

Descriptor Code: 5

CNZZ214I Syntax error: message ID missing

Explanation: Message CNZZ214I is issued during the processing of both MSG statements and SETMF commands containing a MSG= parameter. In the case of a MSG statement, the message ID is either not present or is separated from the MSG specification by more than one blank. In the case of a MSG= parameter, the message ID is either not present or is separated from the MSG= specification by a blank.

System Action: The MSG Parmlib statement or SETMF command is not processed.

Operator Response: Correct the MSG= specification and re-enter the SETMF command.

System Programmer Response: Correct the MSG Parmlib statement and re-load the MSGFLDxx Parmlib member.

Module: CNZZTDP4

Routing Code: 2

Descriptor Code: 5

CNZZ301I Value of varname is nnnnnnnnn

Explanation: Message CNZZ301I is issued in response to the DISPLAY MSGFLD,MSGTYPE=msgtype,keyword command.

In the message text:

varname

The name of the variable for which the value was requested.

nnnnnnnnn

The value of the requested variable.

System Action: The value of the specified variable is returned.

Operator Response: None.

System Programmer Response: None.

Module: CNZZDVL1, CNZZDVL2, CNZZDVL3, CNZZDVL4

Routing Code:

* The message will be routed back to the console that initiated the associated request.

Descriptor Code: 5

CNZZ302I Invalid value length *errtxt*

Explanation: The CNZZ302I message can be produced by either MSGFLDxx Parmlib member processing or by command processing. During the processing of a REGULAR, ACTION or SPECIFIC MSGFLDxx Parmlib statement, the length of the value provided is either too short or too long. During the processing of a DISPLAY MSGFLD,MSGTYPE= msgtype,keyword or SETMF MSGTYPE=msgtype,keyword=value command, the length of the value provided is either too short or too long.

In the message text:

errtxt

The first 12 characters of the string that is in error.

System Action: The statement or command is not processed.

Operator Response: Correct and re-issue the DISPLAY MSGFLD or SETMF MSGTYPE command.

System Programmer Response: Correct the appropriate MSGFLDxx Parmlib statement and re-load the MSGFLDxx Parmlib member.

Module: CNZZTOKN

Routing Code: 2

Descriptor Code: 5

CNZZ303I Improperly specified value *errtxt*

Explanation: The CNZZ303I message can be produced by either MSGFLDxx Parmlib member processing or by command processing. During the processing of a REGULAR, ACTION or SPECIFIC MSGFLDxx Parmlib statement, the floating point value provided was improperly specified. During the processing of a DISPLAY MSGFLD,MSGTYPE= msgtype,keyword or SETMF MSGTYPE=msgtype,keyword=value command, the floating point value provided was improperly specified.

In the message text:

errtxt

The first 12 characters of the string that is in error.

System Action: The statement or command is not processed.

Operator Response: Correct and re-issue the DISPLAY MSGFLD or SETMF MSGTYPE command.

System Programmer Response: Correct the appropriate MSGFLDxx Parmlib statement and re-load the MSGFLDxx Parmlib member.

Module: CNZZTOKN

Routing Code: 2

Descriptor Code: 5

CNZZ304I Value not in range *errtxt*

Explanation: The CNZZ304I message can be produced by either MSGFLDxx Parmlib member processing or by command processing. During the processing of a REGULAR, ACTION or SPECIFIC MSGFLDxx Parmlib statement, the floating point value provided was too small or too large. During the processing of a DISPLAY MSGFLD,MSGTYPE= msgtype,keyword or SETMF MSGTYPE=msgtype,keyword=value command, the floating point value provided was too small or too large.

In the message text:

errtxt

The first 12 characters of the string that is in error. Floating point values must be in the range 0.000001 to 16777215.0

System Action: The statement or command is not processed.

Operator Response: Correct and re-issue the DISPLAY MSGFLD or SETMF MSGTYPE command.

System Programmer Response: Correct the appropriate MSGFLDxx Parmlib statement and re-load the MSGFLDxx Parmlib member.

Module: CNZZTOKN

Routing Code: 2

Descriptor Code: 5

CNZZ401I Message Flood Automation loading: *membernm*

Explanation: The CNZZ401I message is issued in response to the SET MSGFLD=xx command. Message Flood Automation is attempting to load the requested MSGFLDxx Parmlib member.

In the message text:

membernm

The name of the MSGFLDxx Parmlib member whose loading was requested.

System Action: Loading of the requested MSGFLDxx Parmlib member continues.

Operator Response: None.

System Programmer Response: None.

Module: CNZZPRLB

Routing Code: 2

Descriptor Code: 5

CNZZ402I IEFPRMLB ALLOCATE failed. RC=*returncode* Reason Code=*reasoncode*

Explanation: Message Flood Automation was attempting to read a MSGFLDxx Parmlib member but was unable to allocate the Parmlib data set concatenation. The return code and reason code are from the IEFPRMLB system service that is used to read the MSGFLDxx Parmlib member.

In the message text:

returncode

The return code describing the failure.

reasoncode

The reason code describing the failure.

System Action: Message Flood Automation terminates the attempt to read the requested MSGFLDxx Parmlib member.

Operator Response: Contact your system programmer.

System Programmer Response: Report this problem to IBM Service.

Module: CNZZPRLB

Routing Code: 2

Descriptor Code: 5

CNZZ403I PARMLIB member *membernm* not found.

Explanation: Message Flood Automation was attempting to read a MSGFLDxx Parmlib member but was unable to find the requested member in the Parmlib data set concatenation.

In the message text:

membernm

The name of the MSGFLDxx Parmlib member whose loading was requested.

System Action: Message Flood Automation terminates the attempt to read the requested MSGFLDxx Parmlib member.

Operator Response: Verify that the name of the MSGFLDxx Parmlib member was not entered incorrectly. If the name was correct, contact your system programmer. Otherwise, re-enter the correct name.

System Programmer Response: Verify that the requested MSGFLDxx Parmlib member exists within the Parmlib concatenation.

Module: CNZZPRLB

Routing Code: 2

Descriptor Code: 5

CNZZ404I I/O error encountered reading PARMLIB.

Explanation: Message Flood Automation was attempting to read a MSGFLDxx Parmlib member but encountered an I/O error while attempting to read Parmlib.

System Action: Message Flood Automation terminates the attempt to read the requested MSGFLDxx Parmlib member.

Operator Response: Report this error to your system programmer.

System Programmer Response: Verify that the system has access to the DASD volumes containing the data sets in the Parmlib concatenation and that none of the devices are reporting errors. Consult the SYSLOG for other system error messages produced by this error.

Module: CNZZPRLB

Routing Code: 2

Descriptor Code: 5

CNZZ405I Error encountered while opening PARMLIB.

Explanation: Message Flood Automation was attempting to read a MSGFLDxx Parmlib member but encountered an error while attempting to open a data set in the Parmlib concatenation.

System Action: Message Flood Automation terminates the attempt to read the requested MSGFLDxx Parmlib member.

Operator Response: Report this error to your system programmer.

System Programmer Response: Consult the SYSLOG for other system error messages produced by this error.

Module: CNZZPRLB

Routing Code: 2

Descriptor Code: 5

CNZZ406I Allocation of a PARMLIB data set failed.

Explanation: Message Flood Automation was attempting to read a MSGFLDxx Parmlib member but encountered an error while attempting to allocate a data set in the Parmlib concatenation.

System Action: Message Flood Automation terminates the attempt to read the requested MSGFLDxx Parmlib member.

Operator Response: Report this error to your system programmer.

System Programmer Response: Consult the SYSLOG for other system error messages produced by this error.

Module: CNZZPRLB

Routing Code: 2

Descriptor Code: 5

CNZZ407I PARMLIB data set concatenation failed.

Explanation: Message Flood Automation was attempting to read a MSGFLDxx Parmlib member but was unable to concatenate the data sets in the Parmlib concatenation.

System Action: Message Flood Automation terminates the attempt to read the requested MSGFLDxx Parmlib member.

Operator Response: Report this error to your system programmer.

System Programmer Response: Consult the SYSLOG for other system error messages produced by this error.

Module: CNZZPRLB

Routing Code: 2

Descriptor Code: 5

CNZZ408I PARMLIB reader loading failed.

Explanation: Message Flood Automation was attempting to read a MSGFLDxx Parmlib member but was unable to LOAD the Parmlib reading routine.

System Action: Message Flood Automation terminates the attempt to read the requested MSGFLDxx Parmlib member.

Operator Response: Report this error to your system programmer.

System Programmer Response: Report this error to IBM Service.

Module: CNZZPRLB

Routing Code: 2

Descriptor Code: 5

CNZZ409I PARMLIB reader unable to access PARMLIB.

Explanation: Message Flood Automation was attempting to read a MSGFLDxx Parmlib member but the Parmlib reading routine was unable to access Parmlib.

System Action: Message Flood Automation terminates the attempt to read the requested MSGFLDxx Parmlib member.

Operator Response: Report this error to your system programmer.

System Programmer Response: Report this error to IBM Service.

Module: CNZZPRLB

Routing Code: 2

Descriptor Code: 5

CNZZ410I Message Flood Automation loading of *membernm* complete.

Explanation: Message Flood Automation completed reading the requested MSGFLDxx Parmlib member.

In the message text:

membernm

The name of the MSGFLDxx Parmlib member whose loading was requested.

System Action: Message Flood Automation uses the policy information read in from the requested MSGFLDxx Parmlib member.

Operator Response: None

System Programmer Response: None

Module: CNZZPRLB

Routing Code: 2

Descriptor Code: 5

CNZZ411I IEFPRMLB READMEMBER failed. RC=*returncode* Reason Code=*reasoncode*

Explanation: Message Flood Automation was attempting to read a MSGFLDxx Parmlib member but encountered an error condition. The return code and reason code are from the IEFPRMLB system service that is used to read the MSGFLDxx Parmlib member.

In the message text:

returncode

The return code describing the failure.

reasoncode

The reason code describing the failure.

System Action: Message Flood Automation terminates the attempt to read the requested MSGFLDxx Parmlib member.

Operator Response: Report this error to your system programmer.

System Programmer Response: Report this problem to IBM Service.

Module: CNZZPRLB

Routing Code: 2

Descriptor Code: 5

CNZZ412I IEFPRMLB FREE failed. RC=*returncode* Reason Code=*reasoncode*

Explanation: Message Flood Automation was attempting to read a MSGFLDxx Parmlib member but was unable to free the Parmlib allocation. The return code and reason code are from the IEFPRMLB system service that is used to read the MSGFLDxx Parmlib member.

In the message text:

returncode

The return code describing the failure.

reasoncode

The reason code describing the failure.

System Action: Message Flood Automation terminates the attempt to read the requested MSGFLDxx Parmlib member.

Operator Response: Report this error to your system programmer.

System Programmer Response: Report this problem to IBM Service.

Module: CNZZPRLB

Routing Code: 2

Descriptor Code: 5

CNZZ413I PARMLIB did not close.

Explanation: Message Flood Automation was attempting to read a MSGFLDxx Parmlib member but was unable to close a data set in the Parmlib concatenation.

System Action: Message Flood Automation terminates the attempt to read the requested MSGFLDxx Parmlib member.

Operator Response: Report this error to your system programmer.

System Programmer Response: Consult the SYSLOG for other system error messages produced by this error.

Module: CNZZPRLB

Routing Code: 2

Descriptor Code: 5

CNZZ414I PARMLIB did not unallocate.

Explanation: Message Flood Automation was attempting to read a MSGFLDxx Parmlib member but was unable to unallocate the Parmlib concatenation.

System Action: Message Flood Automation terminates the attempt to read the requested MSGFLDxx Parmlib member.

Operator Response: Report this error to your system programmer.

System Programmer Response: Consult the SYSLOG for other system error messages produced by this error.

Module: CNZZPRLB

Routing Code: 2

Descriptor Code: 5

CNZZ415I Syntax error: *errtxt*

Explanation: During the processing of a DEFAULTCMD or JOB MSGFLDxx Parmlib statement, a syntax error was detected. During the processing of a SET MSGFLD=, SETMF keyword, SETMF MSGTYPE=, DISPLAY MSGFLD,keyword or DISPLAY MSGFLD,MSGTYPE= command, a syntax error was detected.

In the message text:

errtxt

The first 25 characters of the string that is in error.

System Action: The statement or command is not processed.

Operator Response: Correct and re-issue the DISPLAY MSGFLD, SET MSGFLD= or SETMF command.

System Programmer Response: Correct the appropriate MSGFLDxx Parmlib statement and re-load the MSGFLDxx Parmlib member.

Module: CNZZPRLB, CNZZCMXT

Routing Code: 2

Descriptor Code: 5

CNZZ416I Statement sequence error. *errtxt*

Explanation: The CNZZ416I message is issued during the processing of DEFAULT, DEFAULTCMD, JOB and MSG MSGFLDxx Parmlib member statements. One of the following has occurred:

A DEFAULT statement has been encountered before a valid REGULAR, ACTION or SPECIFIC statement was processed. DEFAULT statements must follow the REGULAR, ACTION or SPECIFIC statements that they refer to.

A DEFAULTCMD statement has been encountered before a valid REGULAR or ACTION statement was processed. DEFAULTCMD statements must follow the REGULAR or ACTION statements that they refer to.

A JOB statement has been encountered before a valid REGULAR or ACTION statement was processed. JOB statements must follow the REGULAR or ACTION statements that they refer to.

A MSG statement has been encountered before a valid SPECIFIC statement was processed. MSG statements must follow the SPECIFIC statements that they refer to.

In the message text:

errtxt

The first 25 characters of the statement that is out of sequence.

System Action: The statement is not processed.

Operator Response: None

System Programmer Response: Correct the sequencing of the appropriate MSGFLDxx Parmlib statements and re-load the MSGFLDxx Parmlib member.

Module: CNZZPRLB

Routing Code: 2

Descriptor Code: 5

CNZZ417I Invalid keyword: *errtxt*

Explanation: The CNZZ417I message is issued during the processing of the DEFAULTCMD MSGFLDxx Parmlib member statement. The keyword provided is not a valid keyword.

In the message text:

errtxt

The first 25 characters of the string that is in error. The keyword may have been misspelled or may not be valid for the particular MSGFLDxx Parmlib statement or command.

System Action: The statement is not processed.

Operator Response: None

System Programmer Response: Correct the appropriate MSGFLDxx Parmlib statement and re-load the MSGFLDxx Parmlib member.

Module: CNZZPRLB

Routing Code: 2

Descriptor Code: 5

CNZZ418I CNZZSCHD IEAMSCHD failed RC=*returncode*

Explanation: Scheduling of the reading of a MSGFLDxx Parmlib member has failed. The return code is from the IEAMSCHD system service which is used to schedule the reading of the Parmlib member to a full-function address space.

In the message text:

returncode

The return code describing the failure.

System Action: Message Flood Automation terminates the attempt to read the requested MSGFLDxx Parmlib member.

Operator Response: Report this error to your system programmer.

System Programmer Response: Report this problem to IBM Service.

Module: CNZZSCHD

Routing Code: 2

Descriptor Code: 5

CNZZ419I CNZZPIRB SCHEDIRB failed RC=*returncode*

Explanation: Scheduling of the reading of a MSGFLDxx Parmlib member has failed. The return code is from the SCHEDIRB system service which is used to schedule the reading of the Parmlib member to a full-function address space.

In the message text:

returncode

The return code describing the failure.

System Action: Message Flood Automation terminates the attempt to read the requested MSGFLDxx Parmlib member.

Operator Response: Report this error to your system programmer.

System Programmer Response: Report this problem to IBM Service.

Module: CNZZSCHD

Routing Code: 2

Descriptor Code: -

CNZZ420I CNZZPRMN running in *asname* address space

Explanation: The routine that performs the reading of a MSGFLDxx Parmlib member from a full-function address space has been successfully invoked in that address space to perform the Parmlib member read.

In the message text:

asname

The name of a full-function address space.

System Action: Message Flood Automation continues the process of reading the requested MSGFLDxx Parmlib member.

Operator Response: None

System Programmer Response: None

Module: CNZZPRMN

Routing Code: 2

Descriptor Code: -

CNZZ421I REGULAR JOBTHRESH must be < MSGTHRESH

Explanation: During the reading of a MSGFLDxx Parmlib member, it was found that the REGULAR JOBTHRESH value was not less than the REGULAR MSGTHRESH value.

System Action: Processing of the MSGFLDxx Parmlib member continues.

Operator Response: Use the SETMF MSGTYPE=REGULAR,JOBTHRESH= command to make the REGULAR JOBTHRESH value less than the REGULAR MSGTHRESH value. Contact your system programmer to fix the values in the MSGFLDxx Parmlib member.

System Programmer Response: Change the REGULAR JOBTHRESH value or REGULAR MSGTHRESH value in the MSGFLDxx Parmlib member so that JOBTHRESH < MSGTHRESH and re-load the MSGFLDxx Parmlib member.

Module: CNZZPRLB

Routing Code: 2

Descriptor Code: 5

CNZZ422I ACTION JOBTHRESH must be < MSGTHRESH

Explanation: During the reading of a MSGFLDxx Parmlib member, it was found that the ACTION JOBTHRESH value was not less than the ACTION MSGTHRESH value.

System Action: Processing of the MSGFLDxx Parmlib member continues.

Operator Response: Use the SETMF MSGTYPE=ACTION,JOBTHRESH= command to make the ACTION JOBTHRESH value less than the ACTION MSGTHRESH value. Contact your system programmer to fix the values in the MSGFLDxx Parmlib member.

System Programmer Response: Change the ACTION JOBTHRESH value or ACTION MSGTHRESH value in the MSGFLDxx Parmlib member so that JOBTHRESH < MSGTHRESH and re-load the MSGFLDxx Parmlib member.

Module: CNZZPRLB

Routing Code: 2

Descriptor Code: 5

CNZZ423I SPECIFIC MSGLIMIT must be < MSGTHRESH

Explanation: During the reading of a MSGFLDxx Parmlib member, it was found that the SPECIFIC MSGLIMIT value was not less than the SPECIFIC MSGTHRESH value.

System Action: Processing of the MSGFLDxx Parmlib member continues.

Operator Response: Use the SETMF MSGTYPE=SPECIFIC,MSGLIMIT= command to make the SPECIFIC MSGLIMIT value less than the SPECIFIC MSGTHRESH value. Contact your system programmer to fix the values in the MSGFLDxx Parmlib member.

System Programmer Response: Change the SPECIFIC MSGLIMIT value or SPECIFIC MSGTHRESH value in the MSGFLDxx Parmlib member so that MSGLIMIT < MSGTHRESH and re-load the MSGFLDxx Parmlib member.

Module: CNZZPRLB

Routing Code: 2

Descriptor Code: 5

CNZZ901I Message Flood Automation parameters

Message type	REGULAR	ACTION	SPECIFIC
INTVLTIME =	ssssssss1	ssssssss2	ssssssss3
JOBIMTIME =	tttttttt1	tttttttt2	
JOBTHRESH =	nnnnnnnn1	nnnnnnnn2	
JOBIMTIME =	tttttttt1	tttttttt2	
MSGCOUNT =	nnnnnnnn3	nnnnnnnn4	nnnnnnnn5
MSGIMTIME =			tttttttt3
MSGLIMIT =			nnnnnnnn6
MSGTHRESH =	nnnnnnnn7	nnnnnnnn8	nnnnnnnn9
NUMJOBS =	nnnnnnnn10	nnnnnnnn11	
SYSIMTIME =	tttttttt4	tttttttt5	tttttttt6
[WARNING: REGULAR JOBTHRESH not < MSGTHRESH]			
[WARNING: ACTION JOBTHRESH not < MSGTHRESH]			
[WARNING: SPECIFIC MSGLIMIT not < MSGTHRESH]			

Explanation: Message CNZZ901I is issued in response to the DISPLAY MSGFLD,PARAMETERS command and provides the current values of the Message Flood Automation parameters, based on the built-in defaults, as modified by the REGULAR, ACTION and SPECIFIC statements contained in the currently active MSGFLDxx Parmlib member.

The warning messages only appear if the REGULAR JOBTHRESH value is less than the REGULAR MSGTHRESH value, the ACTION JOBTHRESH value is less than the ACTION MSGTHRESH value, or the SPECIFIC MSGLIMIT value is less than the SPECIFIC MSGTHRESH value.

In the message text:

sssssss1

The REGULAR interval time in seconds.

sssssss2

The ACTION interval time in seconds.

sssssss3

The SPECIFIC interval time in seconds.

tttttt1

The REGULAR job inter-message time in seconds and fractions of a second.

tttttt2

The ACTION job inter-message time in seconds and fractions of a second.

nnnnnnn1

The REGULAR job threshold message count.

nnnnnnn2

The ACTION job threshold message count.

nnnnnnn3

The REGULAR current message count.

nnnnnnn4

The ACTION current message count.

nnnnnnn5

The SPECIFIC current message count.

tttttt3

The SPECIFIC message inter-message time in seconds and fractions of a second.

nnnnnnn6

The SPECIFIC individual message message threshold count.

nnnnnnn7

The REGULAR message threshold count.

nnnnnnnn8

The ACTION message threshold count.

nnnnnnnn9

The SPECIFIC message threshold count.

nnnnnnnn10

The REGULAR maximum number of jobs to be tracked.

nnnnnnnn11

The ACTION maximum number of jobs to be tracked.

ttttttt4

The REGULAR system inter-message time in seconds and fractions of a second.

ttttttt5

The ACTION system inter-message time in seconds and fractions of a second.

ttttttt6

The SPECIFIC system inter-message time in seconds and fractions of a second.

System Action: Message Flood Automation processing continues.

Operator Response: If any of the warning messages appear, you should correct the problem by raising the MSGTHRESH value until it is greater than the JOBTHRESH (or MSGLIMIT) value, or alternatively, by lowering the JOBTHRESH (or MSGLIMIT) value until it is less than the MSGTHRESH value. You can use the SETMF command to change these values immediately or contact your system programmer to have the values changed in the MSGFLDxx Parmlib member. If you change the values with the SETMF command, these changes will only persist until the next SET MSGFLD=xx command is issued or an IPL occurs. Changing the values in the MSGFLDxx Parmlib member will ensure that the values are properly set any time that the MSGFLDxx Parmlib member is reloaded.

System Programmer Response: Change the appropriate parameters and re-load the MSGFLDxx Parmlib member.

Module: CNZZDVL1

Routing Code:

* The message will be routed back to the console that initiated the associated request.

Descriptor Code: 5

CNZZ902I Message rate monitoring ENABLED.

Explanation: The CNZZ902I message is issued in response to the SETMF MONITORON command. The message rate monitoring function is enabled. All counters are zeroed and a new and a new initial timestamp is stored.

System Action: Message Rate Monitoring data will be collected.

Operator Response: None.

System Programmer Response: None.

Module: CNZZCMXT

Routing Code:

* The message will be routed back to the console that initiated the associated request.

Descriptor Code: 5

CNZZ903I Message rate monitoring DISABLED.

Explanation: The CNZZ903I message is issued in response to the SETMF MONITOROFF command. The message rate monitoring function is disabled. The initial timestamp and all counters remain unchanged.

System Action: Message Rate Monitoring data is retained. No new data is gathered.

Operator Response: The message rate monitoring data that has been gathered may be displayed by issuing the DISPLAY MSGFLD,MSGRATE command.

System Programmer Response: None.

Module: CNZZCMXT

Routing Code:

* The message will be routed back to the console that initiated the associated request.

Descriptor Code: 5

CNZZ904I Message Flood Automation DEFAULTS

Message type	REGULAR	ACTION	SPECIFIC
LOG	= yn01	yn02	yn03
AUTO	= yn04	yn05	yn06
DISPLAY	= yn07	yn08	y909
CMD	= yn10	yn11	

RETAIN = yn12 yn13
IGNORE = yn14

Explanation: Message CNZZ904I is issued in response to the DISPLAY MSGFLD,DEFAULTS command and provides the current settings of the Message Flood Automation default actions, based on the built-in defaults, as modified by the DEFAULT actions from the currently active MSGFLDxx Parmlib member.

In the message text:

yn01

The REGULAR logging action, Y or N.

yn02

The ACTION logging action, Y or N.

yn03

The SPECIFIC logging action, Y or N.

yn04

The REGULAR automation action, Y or N.

yn05

The ACTION automation action, Y or N.

yn06

The SPECIFIC automation action, Y or N.

yn07

The REGULAR console display action, Y or N.

yn08

The ACTION console display action, Y or N.

yn09

The SPECIFIC console display action, Y or N.

yn10

The REGULAR command action, Y or N.

yn11

The ACTION command action, Y or N.

yn12

The ACTION message retention action, Y or N.

yn13

The SPECIFIC message retention action, Y or N.

yn14

Whether Message Flood Automation is to completely ignore a message, Y or N.

System Action: Message Flood Automation processing continues.

Operator Response: None.

System Programmer Response: None.

Module: CNZZDVL2

Routing Code:

* The message will be routed back to the console that initiated the associated request.

Descriptor Code: 5

CNZZ905I Message Flood Automation JOB actions

REGULAR	messages	LOG	AUTO	DISPLAY	CMD	RETAIN
JOB	<i>jobname1</i>	<i>yn01</i>	<i>yn02</i>	<i>yn03</i>	<i>yn04</i>	<i>yn05</i>
ACTION	messages	LOG	AUTO	DISPLAY	CMD	RETAIN
JOB	<i>jobname2</i>	<i>yn06</i>	<i>yn07</i>	<i>yn08</i>	<i>yn09</i>	<i>yn10</i>

Explanation: Message CNZZ905I is issued in response to the DISPLAY MSGFLD,JOBS command and provides the current settings of the Message Flood Automation actions for specific jobs, based on the built-in defaults, as modified by the JOB actions from the currently active MSGFLDxx Parmlib member. The REGULAR heading only appears in the message if REGULAR JOB statements were defined. The ACTION heading only appears in the message if ACTION JOB statements were defined. The JOB line is repeated for each REGULAR or ACTION job that was defined.

In the message text:

jobname1

The name of the job for which these actions will be taken.

yn01

The REGULAR logging action, Y or N.

yn02

The REGULAR automation action, Y or N.

yn03

The REGULAR console display action, Y or N.

yn04

The REGULAR command action, Y or N.

yn05

The REGULAR message retention action, Y or N.

jobname2

The name of the job for which these actions will be taken.

yn06

The ACTION logging action, Y or N.

yn07

The ACTION automation action, Y or N.

yn08

The ACTION console display action, Y or N.

yn09

The ACTION command action, Y or N.

yn10

The ACTION message retention action, Y or N.

System Action: Message Flood Automation processing continues.

Operator Response: None.

System Programmer Response: None.

Module: CNZZDVL3

Routing Code:

* The message will be routed back to the console that initiated the associated request.

Descriptor Code: 5

CNZZ906I Message Flood Automation MSG actions

SPECIFIC	messages	LOG	AUTO	DISPLAY	CMD	RETAIN
MSG	<i>messageid</i>	<i>yn01</i>	<i>yn02</i>	<i>yn03</i>	<i>yn04</i>	<i>yn05</i>

Explanation: Message CNZZ906I is issued in response to the DISPLAY MSGFLD,MSGs command and provides the current settings of the Message Flood Automation actions for specific messages, based on the built-in defaults, as modified by the MSG actions from the currently active MSGFLDxx Parmlib member. The SPECIFIC heading only appears in the message if SPECIFIC MSG statements were defined. The MSG line is repeated for each SPECIFIC message that was defined.

In the message text:

messageid

The message ID for which these actions will be taken.

yn01

The SPECIFIC logging action, Y or N.

yn02

The SPECIFIC automation action, Y or N.

yn03

The SPECIFIC console display action, Y or N.

yn04

The SPECIFIC message retention action, Y or N.

yn05

Whether Message Flood Automation is completely ignore this message, Y or N.

System Action: Message Flood Automation processing continues.

Operator Response: None.

System Programmer Response: None.

Module: CNZZDVL4

Routing Code:

* The message will be routed back to the console that initiated the associated request.

Descriptor Code: 5

SYSLOG records

The SYSLOG will contain information about the messages that Message Flood Automation processed. It will contain Message Flood Automation messages about the decisions it made and the actions it took. It will also contain information about the actions taken on individual messages (unless NOLOG was specified).

Each SYSLOG record is prefaced by a two-character record type field.

Valid first characters are:

- N - single-line message
- W - single-line message with reply
 - WTOR messages are not processed by Message Flood Automation.
- M - first line of a multi-line message
 - Message Flood Automation can only react to the first line of a multi-line message, not to any of the label, data or end lines
- L - multi-line message label line
- D - multi-line message data line
- E - multi-line message data/end line
- S - continuation of previous line
- O - LOG command input
- X - non-hardcopy or LOG command source

Valid second characters are:

- C - command issued by operator
 - R - command response message
 - I - internally issued command
 - U - command from unknown console ID (z/OS R8 and above)
-

SYSLOG MPF Flags

Message Flood Automation sets the following MPF Exit (CTXT) request flags:

- IGNORE: No changes of any kind are made. All of the following actions are ignored if specified.
- NOIGNORE: Allows any of the following actions to be taken (default).
- NOLOG and NODISPLAY: CTXTRDTM is set and the message is deleted and will not appear in the log or on a display.
 - If AUTO was requested, the message will be queued to EMCS consoles that have requested automation messages even though the message will not appear on a display and will not be logged in the SYSLOG.
 - RETAIN/NORETAIN are ignored if NOLOG and NODISPLAY are both specified.
- NOLOG and DISPLAY: CTXTRNHC is set and the message is displayed but not logged
- LOG and NODISPLAY: CTXTRHCO is set and the message is logged but will not appear on a display
- LOG and DISPLAY: CTXTRFHC and CTXTROMS are set and the message is both logged and displayed.
- NOAUTO: CTXTRANO is set and the message is not flagged for automation
- AUTO: CTXTRAYS is set and the message is flagged for automation.
- NORETAIN: CTXTRNRT is set and the message, if it is an action message, is not retained by the Action Message Retention Facility (AMRF)
- RETAIN is a no-op.

If you request NOLOG, the message will not appear in the SYSLOG. Otherwise, you can verify that the appropriate actions are being taken by examining the MPF Request Flag field in the SYSLOG. The MPF Request Flag field is an 8-character hexadecimal field located at offset 46 in each SYSLOG record (a zero-relative offset is assumed).

The following information is from the IHAHCLOG member of SYS1.MODGEN, which maps a SYSLOG record.

The following hexadecimal values in the 8-character field are pertinent:

```
xxxxxxxx
8..... - message text changed
4..... - route codes changed
2..... - descriptor codes changed
1..... - message queued to specific active console

.8..... - unused
.4..... - message queued by route codes only
.2..... - console ID was changed
```

```

.1..... - minor lines of MLWTO processed by exit

..8..... - the message was deleted (this never appears!)
..4..... - suppression was overridden (DISPLAY was forced)
..2..... - forced to hardcopy (LOG forced)
..1..... - bypass hardcopy (this never appears!)

...8.... - forced to hardcopy only (LOG and NODISPLAY)
...4.... - broadcast
...2.... - don't broadcast
...1.... - don't retain in AMRF (NORETAIN)

....8... - retain in AMRF (RETAIN)
....4... - retrieval key changed
....2... - 4-byte console ID changed
....1... - message type changed

.....8.. - do not automate (set by MPFLSTxx and NOAUTO)
.....4.. - automate (set by MPFLSTxx)
.....2.. - message issued hardcopy-only
.....1.. - message was undelivered

.....8. - message not serviced by any exit
.....4. - ESTAE error in IEAVX600
.....2. - message not serviced, incompatible request
.....1. - automation requested (AUTO)

.....8 - not queued to any console
.....4 - suppressed by subsystem
.....2 - suppressed by exit (NODISPLAY)
.....1 - suppressed by MPF (set by MPFLSTxx and NODISPLAY)

```

Note that in any column, these hexadecimal values are additive. For example, requesting LOG, NODISPLAY, NOAUTO results in a field containing '00080A03' which is interpreted as:

- message forced to hardcopy-only
- don't automate message
- message issued to hardcopy-only
- message suppressed by exit
- message suppressed by MPF

SYSLOG Message ordering

Message Flood Automation processing is driven by the issuance of a message. As soon as that message is created, it obtains a timestamp, and this occurs before the message is seen by Message Flood Automation. If Message Flood Automation makes a decision based on that message, it interrupts the processing of that message until it has taken whatever action it needs to take. Once that has occurred, processing of the original message by the operating system is then allowed to resume.

In the following example, the *second* IOS100I message caused Message Flood Automation to exceed the REGULAR message threshold. Further processing of the second IOS100I message was then suspended while:

- Message Flood Automation issued its CNZZ002E message
- Message Flood Automation took action against the second IOS100I message (as seen in its MPF flags)

The second IOS100I message was then allowed to continue, causing it to be written to the SYSLOG *after* the CNZZ002E message had been written to it.

```
11:14:58.79 ... 00000010 IOS100I DEVICE 891B BOXED, LAST PATH 75 LOST
11:14:58.91 ... 00000010 CNZZ002E MESSAGE THRESHOLD REACHED FOR JOB NONAME
11:14:58.79 ... 00080A03 IOS100I DEVICE 891C BOXED, LAST PATH 75 LOST
```

Recovery

All working storage acquired by Message Flood Automation is protected by ESTAEs. If an error occurs while dynamic storage is held, the appropriate recovery routine is invoked by the operating system Recovery Termination Manager (RTM). Each recovery routine attempts to put out a message indicating the source of the error and frees the dynamic storage that is held before returning to RTM for normal system error handling.

Dynamic storage is only acquired to process commands, read PARMLIB and perform intensive mode processing, so ESTAE overhead is only incurred in those situations. Normal message processing does not establish ESTAE protection and recovery routines will not be invoked should an error occur during normal message processing.

If the ABEND is in the CNZZVMXT message exit or its subroutines, z/OS disables the entire exit. Depending on the error, it may be possible to re-activate the message exit with a 'K M,UEXIT=Y' command.

If the ABEND is in the CNZZCMXT command exit or its subroutines, z/OS disables the entire exit. Depending on the error, it may be possible to re-activate the command exit with a 'SET MPF=xx' command.

Other Information

Message Flood Automation acquires approximately 2000 bytes of private, above-the-line dynamic storage (subpool 230) when the Message Flood Automation message exit is invoked during intensive mode processing or when the Message Flood Automation command exit is invoked to process a command. No dynamic storage is obtained for normal mode processing. A small amount of private, above-the-line storage (subpool 8) is obtained when the MSGFLDxx PARMLIB member is read.

Message Flood Automation acquires approximately 3400 bytes of above-the-line SQA storage (subpool 241) for use by its global parameters and counters. The SQA area is obtained during Message Flood Automation initialization. This storage persists until explicitly freed by operator command (SETMF FREE).

Terms and Conditions

The Message Flood Automation software has not been tested with ISV software, including message automation products. Before installing and activating the Message Flood Automation function, you will need to assess whether there are possible interactions between Message Flood Automation and any ISV software you run. Use of Message Flood Automation with ISV software may require adjustments to Message Flood Automation policy, ISV policy or both and it is possible that Message Flood Automation cannot be used in conjunction with particular ISV software.

Service is provided by IBM Support, Console Services, Level2.

Installation

Instructions for installing Message Flood Automation

IMPORTANT! If you have already been running Message Flood Automation and you are now installing an updated level of Message Flood Automation, be sure to see "Considerations when migrating from one level to another" *before* deactivating the old version and activating the new version!

Message Flood Automation consists of the following load modules in SYS1.LINKLIB:

- CNZZCMXT
- CNZZVMXT

and the following sample Assembly Language programs in SYS1.SAMPLIB:

- CNZZVXT1
- CNZZVXT2

To use Message Flood Automation, you must:

- Select one of the sample programs from SYS1.SAMPLIB
- Assemble it with the High-Level Assembler
- Link it to CNZZVMXT with the Linkage Editor or Binder

The *SA22-7593 z/OS MVS Installation Exits* book provides additional information on the installation and use of message and command exits.

Criteria for selecting either CNZZVXT1 or CNZZVXT2

You can use either CNZZVXT1 or CNZZVXT2.

If you do not already have an IEAVMXIT message exit, CNZZVXT1 is the simplest to use, but you can use CNZZVXT2, without change, if you wish. CNZZVXT2 has additional complexity that CNZZVXT1 does not.

If you already have an IEAVMXIT, you should use CNZZVXT2 and either integrate your existing IEAVMXIT function into it, or use CNZZVXT2 as an example of how to place the invocation of Message Flood Automation into your existing IEAVMXIT message exit. See "[Customization](#)" in the Appendix for additional interface information.

Assembling CNZZVXT1 or CNZZVXT2

You can use the High-Level Assembler to assemble either CNZZVXT1 or CNZZVXT2. The following JCL uses the High-Level Assembler to assemble CNZZVXT1 and place it into an existing data set userid.SAMPLIB.OBJ.

```
//ASM      EXEC   PGM=ASMA90,REGION=6144K,
//          PARM='OBJECT,NODECK,XREF(SHORT),LIST(133),ALIGN,MACHINE(X
//          ZS-2,LIST),GOFF'
//SYSLIB   DD   DSN=SYS1.MACLIB,DISP=SHR
//          DD   DSN=SYS1.MODGEN,DISP=SHR
//SYSUT1   DD   UNIT=SYSDA,SPACE=(CYL,(20,5)),DSN=&SYSUT1
//SYSUT2   DD   UNIT=SYSDA,SPACE=(CYL,(20,5)),DSN=&SYSUT2
//SYSUT3   DD   UNIT=SYSDA,SPACE=(CYL,(20,5)),DSN=&SYSUT3
//SYSPRINT DD   SYSOUT=A
//SYSPUNCH DD   DUMMY
//SYSLIN   DD   DSN=userid.SAMPLIB.OBJ(CNZZVXT1),DISP=OLD
//SYSIN    DD   DSN=SYS1.SAMPLIB(CNZZVXT1),DISP=SHR
```

Linking CNZZVXT1 or CNZZVXT2 with CNZZVMXT

You can use the following sample JCL to link either CNZZVXT1 or CNZZVXT2 to CNZZVMXT. The result of the link will be a part named IEAVMXIT in SYS1.LINKLIB.

For SYSLMOD, you may use any data set in the LINKLIB concatenation.

```
//LKED     EXEC   PGM=IEWL,REGION=0M,
//          PARM='XREF,LIST,RENT,REFR,REUS,AC(0)'
//SYSPRINT DD   SYSOUT=A
//BASE     DD   DSN=SYS1.LINKLIB,DISP=SHR
//SYSUT1   DD   UNIT=SYSDA,SPACE=(CYL,(1,1))
//SYSLMOD  DD   DSN=SYS1.LINKLIB,DISP=SHR
//SYSLIN   DD   DSN=userid.SAMPLIB.OBJ(CNZZVXT1),DISP=SHR
//          DD   *
//          SETCODE AC(0)
//          INCLUDE BASE(CNZZVMXT)
//          ENTRY  IEAVMXIT
//          NAME   IEAVMXIT(R)
```

For z/OS R9: if you receive IEW2609W messages from the binder, remove the REFR parameter from the PARM= specification.

CNZZVXT1, CNZZVXT2, CNZZVMXT and CNZZCMXT are all AMODE=31 and RMODE=ANY.

Creating an SMP/E ++USERMOD

As an alternative to manually assembling and linking CNZZVXT1 or CNZZVXT2 with CNZZVMXT, you can use SMP/E to perform the operation for you, creating an SMP/E ++USERMOD.

IBM recommends using the SMP/E ++USERMOD approach since that method will allow SMP/E to automatically determine if and when your user exit must be relinked. It is necessary to keep both the user exit and the IBM load modules synchronized, and SMP/E is best positioned to do this automatically for you.

```
++USERMOD (xxxxxxx) REWORK(yyyyddd)
/* This sample usermod can be used to install the common MPF
   message exit, IEAVMXIT, created for Message Flood
   Automation. Change as indicated below.

   Change xxxxxxx and yyyyddd to an appropriate usermod
   name and a rework date

   When directed by IBM PTF HOLD ACTIONS, you may have to
   change this usermod.

   For additional information on SMP/E and usermods, refer
   to the following publications:

       SMP/E for z/OS User's Guide SA22-7773
       SMP/E Reference SA22-7772
*/ .
++VER (Z038) FMID (HBBrrrr)
PRE (ptf#).                               /* Change rrrr to an appropriate
                                           release level

                                           Place the appropriate PTF
                                           number for OA17514 in the
                                           PRE statement

                                           Note: Unless otherwise indicated
                                           within IBM PTF HOLD ACTIONS,
                                           including the PTF for OA17514 in
                                           the PRE statement is sufficient.
                                           Placing the PTF of a follow on
                                           Message Flood Automation apar,
                                           though not required, would work
                                           as well since that PTF would be
                                           linked to the PTF of OA17514 via
                                           the SMP/E logic.                               */

++JCLIN CALLLIBS.
//SG35 EXEC LINKS,
//*****
/* Change NAME=userlib below to point to the desired user exit *
/* library in the linklist                                     *
//*****
// PARM='NCAL,LIST,XREF,LET,RENT,REFR',
// UNIT='3380',SER=MVSRS1,N=SYS1,NAME=userlib,P1=' ',
// MOD=,P2=' ',OBJ=MACLIB,CLASS=N
//AOSC5 DD DISP=SHR,VOLUME=(,RETAIN),DSN=SYS1.AOSC5
```

```

//SYSLIN DD *
INCLUDE SYSPUNCH(IEAVMXIT)
INCLUDE AOSC5(CNZZVMXT)
INCLUDE AOSC5(CNZZVMES)
INCLUDE AOSC5(CNZZVMIM)
INCLUDE AOSC5(CNZZCKRT)
INCLUDE AOSC5(CNZZRIMN)
INCLUDE AOSC5(CNZZAIMN)
INCLUDE AOSC5(CNZZSIMN)
INCLUDE AOSC5(CNZZRACT)
INCLUDE AOSC5(CNZZAACT)
INCLUDE AOSC5(CNZZINIT)
INCLUDE AOSC5(CNZZRIOF)
INCLUDE AOSC5(CNZZAIOF)
INCLUDE AOSC5(CNZZSIOF)
ENTRY IEAVMXIT
NAME IEAVMXIT(R) RC=0
++SRC(IEAVMXIT) DISTLIB(dlibloc) /* Change DISTLIB(dlibloc) to point
                                   to the desired dlib location of
                                   source code

                                   Insert source code below,
                                   following the ++SRC statement */

        PRINT ON,NOGEN
        TITLE 'IEAVMXIT - sample for users w/o an existing IEAVMXIT'
***START OF SPECIFICATIONS*****
*                                                                           *
* *01* MODULE NAME: CNZZVXT1                                           *
*                                                                           *
*                                                                           *
*-----*
IEAVMXIT CSECT
IEAVMXIT AMODE 31
IEAVMXIT RMODE ANY
*
*

```

Using CNZZCMXT

1. Add a **.CMD USEREXIT(CNZZCMXT)** statement to the MPFLSTxx PARMLIB member that is being used. If you already have one or more command exit(s) specified, just add CNZZCMXT to the existing specification.

```

before:  .CMD USEREXIT(YOUREXIT)
after:   .CMD USEREXIT(YOUREXIT,CNZZCMXT)

```

You can add CNZZCMXT either before or after your existing exit(s). The .CMD statement supports a maximum of 6 exit specifications.

2. An LLA refresh may be necessary to make the exit available after it has been link-edited. To do so, issue a **F LLA,REFRESH** command from a z/OS console.
 - o If you make the command exit available by changing the libraries referred to in the LINKLIST concatenation, you must issue a
 - **SETPROG LNKLST,UPDATE,JOB=*MASTER*** command to cause the Master Scheduler address space to use the new LINKLIST

- concatenation. The SET MPF= command runs in the Master Scheduler address space.
 - o *z/OS Initialization and Tuning Reference*: There is some risk associated with the SETPROG command (ABEND106 errors could result) and therefore it is recommended that this command be used only when necessary to prevent an IPL.
3. Issue a **SET MPF=xx** command to activate the updated MPFLSTxx member containing the .CMD statement. The exit should be automatically loaded and activated.

Using IEAVMXIT

1. Add a UEXIT(Y) parameter to the INIT CONSOLxx statement. This is not strictly necessary since the default is UEXIT(Y), but explicitly specifying UEXIT(Y) can serve to remind you that a user-exit is in use. Be sure that your installation has not specified UEXIT(N) since this will prevent IEAVMXIT from being activated.
2. An LLA refresh may be necessary to make the exit available after it has been assembled and link-edited. To do so, issue a **F LLA,REFRESH** command from a z/OS console.
 - o If you make the message exit available by changing the libraries referred to in the LINKLIST concatenation, you must issue a
 - **SETPROG LNKLIST,UPDATE,JOB=CONSOLE** command to cause the Console address space to use the new LINKLIST concatenation. The K M command runs in the Console address space.
 - o *z/OS Initialization and Tuning Reference*: There is some risk associated with the SETPROG command (ABEND106 errors could result) and therefore it is recommended that this command be used only when necessary to prevent an IPL.
3. Issue a **K M,UEXIT=Y** command to activate IEAVMXIT/CNZZVMXT.

IEAVMXIT and Message Flood Automation can be disabled at any time by issuing a **K M,UEXIT=N** command.

The status of IEAVMXIT and Message Flood Automation can be determined at any time by issuing a **K M,REF** command. You can also find out about IEAVMXIT status by issuing the **D MPF,MSG** command.

Providing a MSGFLDxx member of PARMLIB

You need to provide a MSGFLDxx member in PARMLIB.

1. See the sample MSGFLDxx member shown in "Sample PARMLIB specification". You should provide a MSGFLDxx member similar to it and place the member into a data set in the PARMLIB concatenation.
2. You may have as many MSGFLDxx PARMLIB members as you like but Message Flood Automation only supports one member being active at a time. Message Flood Automation processing requires that the MSGFLDxx suffix (the "xx") be alphabetic or numeric. National or other special characters are not supported.

Considerations when migrating from one level to another

If you have already been running Message Flood Automation and you are now installing a newer level, please make sure that you do the following in addition to the instructions in "[Installation](#)".

- Prior to installing and activating a new level of Message Flood Automation, the old version must be deactivated and removed from the system(s) properly.
- Make sure that you install and activate newer levels of both IEAVMXIT/CNZZVMXT and CNZZCMXT at the same time. CNZZVMXT and CNZZCMXT both map a shared, common data area. To ensure that both CNZZVMXT and CNZZCMXT are consistent in their mapping of this area, you **MUST** install and activate newer levels of both at the same time.
- Make sure that you have freed the old Message Flood Automation shared common data area using the SETMF FREE command *before* you activate the new level of Message Flood Automation using the K M,UEXIT=Y and SET MPF=xx commands. (If you are coming from a 1.2.x level of Message Flood Automation, use the SET MSGFLD=FREE command to free the shared common data area).

The order in which you shutdown the exits and reactivate them is critical to the success of this operation:

1. Issue a **SETMF OFF** command to make sure that Message Flood Automation processing has been disabled. (If you are coming from a 1.2.x level of Message Flood Automation, use the SET MSGFLD=OFF command to disable Message Flood Automation).
2. Issue a **SETMF MONITOROFF** command to make sure that Message Rate Monitoring has been disabled. (If you are coming from a 1.2.x level of Message Flood Automation, use the SET MSGFLD=MONITOROFF command to disable Message Rate Monitoring).
3. Issue a **SETMF FREE** command. (If you are coming from a 1.2.x level of Message Flood Automation, use the SET MSGFLD=FREE command to free the shared common data area).

Issuing this command will free the common data area that is shared by the message and command exit parts of Message Flood Automation. This operation also sets a flag that will cause the anchor for the common data area to be entirely removed the next time that the message exit is terminated. The flag also prevents the command exit from remembering the address of the common data area.

4. Issue a **SET MPF=xx** command which loads a version of MPFLSTxx *which does not contain* a **.CMD USEREXIT** definition for the Message Flood Automation command exit.

This will cause the command exit to be terminated and not be reloaded. The message exit should be reloaded *first* since it establishes the common data area and its anchor.

5. Issue a **K M,UEXIT=N** command.

This command will terminate the Message Flood Automation message exit and not reload it.

6. **At this point, you should update your system library with the new level of Message Flood**

Automation and do an LLA refresh if necessary.

- Make sure that the **OLD** level of *both* the message exit (IEAVMXIT) and the command exit (MFACMDXT) have been removed.
 - Make sure that the **NEW** level of *both* the message exit (IEAVMXIT) and the command exit (CNZZCMXT) are present.
 - If you make the new level of both exits available by changing the libraries referred to in the LINKLIST concatenation, you must issue the following commands:
 - **SETPROG LNKLST,UPDATE,,JOB=CONSOLE**
 - **SETPROG LNKLST,UPDATE,,JOB=*MASTER***to cause the Console address space and the Master Scheduler address space to use the new LINKLIST concatenation. The K M command runs in the Console address space; the SET MPF= command runs in the Master Scheduler address space.
 - *z/OS Initialization and Tuning Reference*: There is some risk associated with the SETPROG command (ABEND106 errors could result) and therefore it is recommended that this command be used only when necessary to prevent an IPL.
7. Issue a **K M,UEXIT=Y** command to load the new level of Message Flood Automation message exit.
 8. Issue a **SET MPF=xx** command which loads a version of MPFLSTxx *which contains a .CMD USEREXIT* definition for the new level of the Message Flood Automation command exit.
 9. Optionally issue a **SETMF ON** command to enable Message Flood Automation so that the enablement status can be verified by the following command.
 10. Issuing a **D MF,STATUS** command should show the new level of Message Flood Automation is now active (and enabled if the previous SETMF ON command was issued).

A system ABEND X'077-039' from the message exit or error message CNZZ013I from the command exit are an indication that these steps were not followed correctly.

MSGFLDxx PARMLIB considerations

If you are coming from a level of Message Flood Automation prior to the 1.4.0 level, you will need to make the following alterations:

- In the SPECIFIC message section of your MSGFLDxx PARMLIB member(s), add a specification for the **MSGLIMIT** parameter. The MSGLIMIT value must be less than the MSGTHRESH value (see below).
 - **MSGLIMIT** serves the same purpose for SPECIFIC messages as the **JOBTHRESH** parameter does for ACTION and REGULAR messages.

If you are coming from a level of Message Flood Automation prior to the 2.0.0 level, you will need to make the following alterations:

- Within the ACTION and REGULAR message sections of your MSGFLDxx PARMLIB member (s), adjust your MSGTHRESH and JOBTHRESH parameters so that JOBTHRESH < MSGTHRESH. Failure to make this change will result in a CNZZ421I and/or CNZZ422I error message being issued when the MSGFLDxx PARMLIB member is loaded. Warning statements will also be displayed as part of the CNZZ901I message when the D MF,PARAMETERS

command is issued.

- Within the SPECIFIC message section of your MSGFLDxx PARMLIB member(s), adjust your MSGTHRESH and MSGLIMIT parameters so that MSGLIMIT < MSGTHRESH. Failure to make this change will result in a CNZZ423I error message being issued when the MSGFLDxx PARMLIB member is loaded. A warning statement will also be displayed as part of the CNZZ901I message when the D MF,PARAMETERS command is issued.

Users of earlier Message Flood Automation versions should re-evaluate their message rates and threshold settings. The values and settings from a prior version may not be optimal or as effective for the z/OS version.

Instructions for initializing Message Flood Automation.

The CNZZCMXT command exit routine will be automatically loaded into storage once it has been link-edited into a data set in the LINKLIB concatenation and an LLA refresh (or IPL) has been performed.

- If you make the command exit available by changing the libraries referred to in the LINKLIST concatenation, you must issue a
 - **SETPROG LNKLST,UPDATE,JOB=*MASTER*** command to cause the Master Scheduler address space to use the new LINKLIST concatenation. The SET MPF= command runs in the Master Scheduler address space.
- *z/OS Initialization and Tuning Reference*: There is some risk associated with the SETPROG command (ABEND106 errors could result) and therefore it is recommended that this command be used only when necessary to prevent an IPL.

The CNZZCMXT command exit routine will be automatically activated when a **SET MPF** command is issued that results in processing the MPFLSTxx member that contains the command exit routine's .CMD specification. Once activated, the command exit does nothing until it receives a **SET MSGFLD**, **SETMF** or **DISPLAY MSGFLD** command of some kind. During a normal IPL, the command exit will be loaded and activated shortly *before* operating system command processing becomes active. **You cannot use a COMMNDxx member of PARMLIB to issue SET MSGFLD commands to load the MSGFLDxx PARMLIB member because COMMNDxx processing occurs prior to the availability of the system services required to read PARMLIB.** Also, you cannot use COMMNDxx to issue the SETMF ON command to enable Message Flood Automation because COMMNDxx processing occurs before the CNZZCMXT command exit is automatically loaded by the system during IPL.

The IEAVMXIT/CNZZVMXT message exit routine will be automatically loaded into storage once it has been link-edited into SYS1.LINKLIB and an LLA refresh (or IPL) has been performed.

- If you make the message exit available by changing the libraries referred to in the LINKLIST concatenation, you must issue a
 - **SETPROG LNKLST,UPDATE,JOB=CONSOLE** command to cause the Console address space to use the new LINKLIST concatenation. The K M command runs in the Console address space.
- *z/OS Initialization and Tuning Reference*: There is some risk associated with the SETPROG command (ABEND106 errors could result) and therefore it is recommended that this command be used only when necessary to prevent an IPL.

The IEAVMXIT/CNZZVMXT message exit routine will be automatically activated when a **K M,UEXIT=Y** command is issued. Once activated, the IEAVMXIT/CNZZVMXT message exit obtains and anchors the shared SQA storage but does nothing until the CNZZCMXT command exit receives and processes a **SET MSGFLD** or **SETMF** command of some kind.

Issue a **SET MSGFLD=00** command from a z/OS console to cause the Message Flood Automation PARMLIB member MSGFLD00 to be read.

- The reading of the PARMLIB member will occur in the DUMPSRV address space.
- You should see a message indicating that PARMLIB member MSGFLD00 is being loaded and a second message message indicating that PARMLIB member MSGFLD00 was read successfully.

To activate Message Flood Automation processing, issue a **SETMF ON** command.

- You should see a message indicating that Message Flood Automation is enabled.

Instructions for shutting down Message Flood Automation.

Issue a **SETMF OFF** command from a z/OS console to logically disable Message Flood Automation.

- You should see a message indicating that message flood automation was disabled.
- You can logically re-enable message flood automation by issuing a **SETMF ON** command from a z/OS console.

You can cause the storage used by message flood automation in SQA to be freed by issuing a **SETMF FREE** command. (This command can only be issued after a **SETMF OFF** command has been issued).

You can totally disable message flood automation by issuing a **K M,UEXIT=N** command from a z/OS console. (This command should be issued only after the **SETMF OFF** and **SETMF FREE** commands have been issued).

The CNZZCMXT command exit can only be disabled by disabling all of MPF processing by issuing a **SET MPF=NO** command. This should only be done as a last resort since message processing will be curtailed and it is very likely that the consoles will become flooded with message traffic. To avoid this, you should have an MPFLSTxx member with the .CMD entry specified and an MPFLSTyy member that does not have the .CMD entry specified but has everything else. Then you can quickly issue SET MPF=NO to disable MPFLSTxx and immediately issue SET MPF=yy to enable MPFLSTyy.

- You should not disable the CNZZCMXT command exit unless you have previously disabled the IEAVMXIT/CNZZVMXT message exit. Once CNZZCMXT is disabled, there is no way to disable IEAVMXIT/CNZZVMXT except through the **K M,UEXIT=N** command, which can be disruptive.

Both CNZZCMXT and IEAVMXIT/CNZZVMXT can be replaced while the system is running by replacing either in LINKLIB and issuing an LLA refresh.

- If you replace both exits by changing the libraries referred to in the LINKLIST concatenation, you must issue
 - **SETPROG LNKLST,UPDATE,JOB=CONSOLE**
 - **SETPROG LNKLST,UPDATE,JOB=*MASTER***
 commands to cause the Console address space and the Master Scheduler address space to use the new LINKLIST concatenation. The K M command runs in the Console address space; the SET MPF= command runs in the Master Scheduler address space.
- *z/OS Initialization and Tuning Reference*: There is some risk associated with the SETPROG command (ABEND106 errors could result) and therefore it is recommended that this command be used only when necessary to prevent an IPL.

A K M,UEXIT=Y command is required to reload IEAVMXIT/CNZZVMXT; a SET MPF= command is required to reload CNZZCMXT.

The SQA data area used by both CNZZCMXT and IEAVMXIT/CNZZVMXT is persistent unless explicitly freed using a **SETMF FREE** command. Both CNZZCMXT and IEAVMXIT/CNZZVMXT are designed to re-attach to a pre-existing SQA data area.

The state of message flood automation can always be queried (once the **SET MPF=** command has been issued to load the CNZZCMXT exit) using the **DISPLAY MSGFLD,STATUS** command. The short form **D MF,STATUS** is also supported.

Appendix

Operator command changes from Message Flood Automation

1.2.x

The **MSGFLD** parameter on the **DISPLAY MSGFLD** command may be abbreviated **MF** allowing commands of the form: **DISPLAY MSGFLD**, **DISPLAY MF**, **D MSGFLD** and **D MF**.

To conform with other system command usage, most forms of the **SET MSGFLD** command have been changed to **SETMF**.

- The **SET MSGFLD=ON** command becomes the **SETMF ON** command.
- The **SET MSGFLD=OFF** command becomes the **SETMF OFF** command.
- The **SET MSGFLD=FREE** command becomes the **SETMF FREE** command.
- The **SET MSGFLD=MONITORON** command becomes the **SETMF MONITORON** command.
- The **SET MSGFLD=MONITOROFF** command becomes the **SETMF MONITOROFF** command.

The **T** abbreviation for **SET** is only supported for the **SET MSGFLD=xx** PARMLIB member loading command. There is no abbreviation for **SETMF**.

The **MSGTYPE=** keyword must be specified to identify a *message type* parameter. In the past, *msgtype* was a positional parameter.

- Commands such as **SET MSGFLD,REGULAR,JOBTHRESH=value** become **SETMF MSGTYPE=REGULAR,JOBTHRESH=value**.

The **JOB=** keyword must be specified to identify a *job identifier* parameter. In the past, **JOB** was a positional parameter.

- Commands such as **SET MSGFLD,REGULAR,JOB,jobname=action** become **SETMF MSGTYPE=REGULAR,JOB=jobname,action**.

The **MSG=** keyword must be specified to identify a *message identifier* parameter. In the past, **MSG** was a positional parameter.

- Commands such as **SET MSGFLD,SPECIFIC,MSG,msgid=action** become **SETMF MSGTYPE=SPECIFIC,MSG=msgid,action**.

Operator command changes from Message Flood Automation 1.3.x/1.4.x

The syntax of Message Flood Automation commands is unchanged from the Message Flood Automation 1.3.x and 1.4.x levels. The messages produced by the commands are also unchanged except for the message ID prefix which is now CNZZ rather than MSGF. The numbering and text of the messages remain the same.

Customization

The following section is for customers who already have an IEAVMXIT message exit and need to either integrate their function with Message Flood Automation or invoke Message Flood Automation from their IEAVMXIT.

Customers who do not already have an IEAVMXIT should skip this section.

The Message Flood Automation CNZZVMXT message processing routine expects almost the same input parameters as IEAVMXIT and sets the same return codes. You must ensure that your IEAVMXIT passes the expected input to CNZZVMXT.

On entry, CNZZVMXT expects the following:

- Register 0 contains a pointer to a *persistent* 4-byte field that can be used by CNZZVMXT to anchor its control blocks.

- If the installation's IEAVMXIT does not use the CTXTUIDA data area (this is the 8-byte data area provided by the system and pointed to by CTXTIWKP) to anchor its own control block, you can have register 0 point to CTXTUIDA (the value in register 0 will be the same as in CTXTIWKP). Note that CTXTUIDA will be zero the first time that it is passed into IEAVMXIT by the system; on subsequent invocations of IEAVMXIT, the system will pass in the *same* CTXTUIDA area, which will now contain a pointer to the Message Flood Automation data structures.
 - If the installation's IEAVMXIT does use CTXTUIDA to anchor its own control block, you should provide a 4-byte field within your control block as an anchor for CNZZVMXT's control block and have register 0 point to that field on entry to CNZZVMXT. This 4-byte field must be zero the first time that you invoke CNZZVMXT. Message Flood Automation is expecting that you will point to the *same*, unchanged, 4-byte area on each subsequent call to CNZZVMXT since this area will now contain a pointer to the Message Flood Automation data structures.
 - If you are having CNZZVMXT return to the caller of IEAVMXIT (not to IEAVMXIT), you must store the contents of register 0, as received from the system, into the register save area (at offset 24) pointed to by register 13. CNZZVMXT assumes that this has been done and will not save the altered register 0 into the save area. However, on exit it will restore register 0 from the save area, restoring register 0 as it came into IEAVMXIT.
- Register 1 contains a pointer that points to a pointer which points to the CTXT
 - Register 13 contains a pointer that points to a standard 18 fullword register save area
 - Register 14 contains a pointer to the return location
 - Register 15 contains a pointer to the CNZZVMXT entry point

On exit, CNZZVMXT returns the following:

- Register 13 contains a pointer to the standard 18 fullword register save area that it was passed on entry
- Register 14 contains a pointer to the return location
- Register 15 contains a return code

You can code your invocation of CNZZVMXT so that CNZZVMXT will return to your IEAVMXIT program when it is finished processing the message. If you do so, you must ensure that your IEAVMXIT program passes the return code set by CNZZVMXT back to the caller of IEAVMXIT. This is referred to as a "nested" invocation of CNZZVMXT.

```

:           Reg 0 points to 4-byte anchor field.
:           Reg 1 points to addr of CTXT.
:           Reg 13 points to a save area that is
:           not the one coming into IEAVMXIT
L    15,=V(CNZZVMXT)  Load entry-point addr of CNZZVMXT.
BASR  14,15          Invoke CNZZVMXT and return here.
:           Reg 15 contains a return code.
```

See sample program **CNZZVXT2** in SYS1.SAMPLIB.

NOTE: CNZZVMXT uses the "unused" fullword at offset 0 in the register save area pointed to by register 13, so don't attempt to use this fullword yourself. Also, be sure to backward and forward chain any new register save area that you provide to the register save area pointed to by register 13 on entry to your IEAVMXIT.

You can also program your IEAVMXIT so that it invokes CNZZVMXT but CNZZVMXT returns to the caller of IEAVMXIT, *not* back to your IEAVMXIT. This latter mechanism may be desirable if your IEAVMXIT does not obtain and free dynamic storage and you do not want to do so to provide a new register save area for CNZZVMXT. This is referred to as a "daisy-chained" invocation of CNZZVMXT.

```
      :                               Reg 0 points to 4-byte anchor field.
      :                               Reg 1 points to addr of CTXT.
      :                               Reg 13 points to save area that
      :                               reg 13 pointed to coming into
      :                               IEAVMXIT.
      :                               Reg 14 points to return point in
      :                               caller of IEAVMXIT.
L     15,=V(CNZZVMXT)  Load entry-point addr of CNZZVMXT.
BR    15              Invoke CNZZVMXT and return to system.
```

See sample program **CNZZVXT1** in SYS1.SAMPLIB.

It may be necessary for some messages to not go through message flood processing. This may be achieved by providing extra front-end code in your IEAVMXIT module or by specifying another MPF exit for this message. The alternative exit may be a dummy (BR14) module. This may be useful where messages are known to be required for automation or operator processing. →