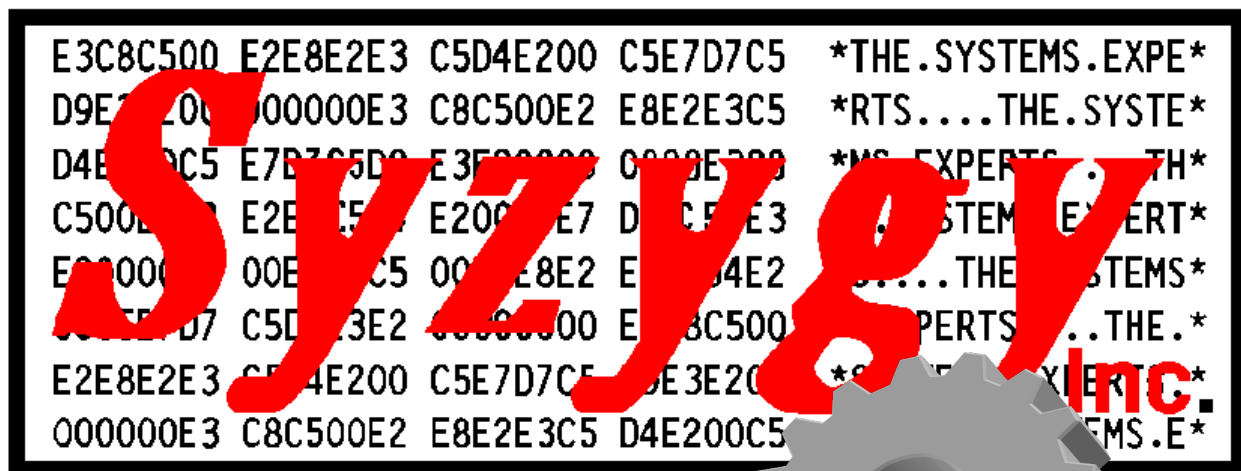


[Version 6]
January 23, 2023

SyzMPF/z



[Installation and User's Guide]

Script Based z/OS Console Message system automation facility

Revision History

Version	Date	Revision Description
6.0.1	1/23/2023	<p>Complete re-write of the internal subroutine modules to sue a faster and even less resource intensive code set. This also makes it easier to add commands over time.</p> <p>Dynamic Variable support added (see # commands)</p> <p>WTO's now fully support upper lower mixed case.</p> <p>Add support for sending sms text from the console or script that doesn't need the message area in the text.</p> <p>IF STRING now uses flexible substring settings.</p> <p>IF #variable support added.</p> <p>New delimiters supported: and =</p> <p>Fixed error where sometimes parsed words would end up starting with a blank space.</p> <p>SyzMPFz now shows the last time used at script replacement task start.</p> <p>SyzMPFz now gets the time zone automatically from the system, no need to specify it in the startup parms anymore. All that needs to be done is reload the scripts via 'S SYZMPFZ" and the new time zone will be picked up.</p>
5.9.1	5/19/2021	<p>Fixed error in Date and Time fields of Query All command</p> <p>Added comments in this manual to show when to use SUPPRESS and when to let IBM's MPF work (see NOCONSOLE).</p>
5.9	2/29/2020	<p>enhanced IFWORDS or IFANYWORD to support generics (% or *)</p> <p>Enhanced DOW processing to support the increment of the day that it is of the month, i.e. the second Tuesday.</p> <p>Added support for the SyzMAIL/s heartbeat so that email won't be sent if the SyzEMAIL/z product is either not licensed or currently inactive. There is also a new startup parm that will allow the site to ignore this feature.</p> <p>Enhanced to allow the ability to send ANY sysout DSID of the task that is being processed. This adds to the previous ability to send any JES dataset to make it so that any part of the task's output may be sent via email or SMS text. There is a current maximum of 50 separate individual sysouts per Task that can be sent in a single email, although you can instead send the entire task output.</p>

		The maximum number of lines that can be attached has been increase to 99,999. There is no logical reason for SyzMPF/z to limit the attached lines except that most sites limit the amount of data allowed in a single email. The previous number was 9,999 lines.
5.8	4/10/2019	<p>Fixes:</p> <p>AUTO(SUPPRESS) - failed on some older hardware</p> <p>AUTO(color) - failed on some older hardware</p> <p>AUTO(NOSYSLOG) - Failed on sole older hardware</p> <p>New Features:</p> <p>AUTO(REVERSEV) - added ability to change messages to reverse video in MPFLIST instead of script</p> <p>AUTO(UNDERLIN) - added ability to underline messages from MPFLIST instead of script</p> <p>AUTO(color) - allows changing of the message color to RED, BLUE, PINL, YELLOW, BLUE, GREEN or WHITE, or TURQuoise from MPFlst. (already worked from script).</p>
5.7.5	7/5/2018	<p>New PARSEDELIMITER command. The default is still a blank, but you can now set a delimiter of ".", "'", "-'" or "/" before PARSE2WORDS to parse the message in new ways. You can set the delimiter and execute PARS2WORDS as many times as you wish to set the &Wnn values as may be deemed necessary.</p> <p>New STRIP WORD command to strip the &WORD variable of all occurrences of up to 4 characters</p> <p>New STRIP STRING command to strip the &WORD variable of all occurrences of up to 4 characters</p> <p>New INCVAR command to increment a persistent or non-persistent numeric variable by '1 (default) or any other increment (up to 9,999,999,999).</p> <p>New DECVAR command to decrement a persistent or non-persistent numeric variable by '1 (default) or any other increment (can't go below zero).</p> <p>New Email sub-parameter "MSGONLY" does not attach the JOBNUMBER and JOBNNAME to the email output.</p> <p>The Tasks TCP and SRB time have been added to MAXCC message output in emails created by SyzMPF/z</p> <p>The GET WORD command was changed to allow use of "@" starting point instead of just STRing to make it easier to get a "part" of the word instead of the entire word. The same was done to the GET STRING command.</p> <p>Added new QUERY WORDS command to display all of the "words" in a message. This is especially handy when you have changed the delimiter to something other than a blank so that you can see what the new words are now set to.</p>

5.7.1	1/23/2018	Fixed minor errors in Email processing if EMAIL with no other options was selected. Fixed rounding calculations of maximum ECSA size actually used. Fixed several spelling errors in some (many) of the new messages
5.6	8/23/2017	SCRIPTS can now be any length (previously was 100 lines max). SCRIPT datasource area in ECSA now can be changed by startup parameter of the Syzygy SubSystem (ECSAMAX). SCRIPT datasource now provides information on the size of the datasource as well as how much is actually USED and WASTED. ECSAMAX global parameter was added to control how much ECSA space to set aside for script usage GET STRING updates <ul style="list-style-type: none"> • GET STRING also accepts GET STR • GET STRING allows @ for begin string location • GET STRING allows 1 or two byte begin location • GET STRING allows 1 or 2 byte length (up to 48) • GET STRING defaults to MAX length if not specified or if specified as greater than 48 bytes GET WORD updates <ul style="list-style-type: none"> • GET WORD allows 1 or 2 byte "WORD" number • GET WORD now allows Substring of the WORD • GET WORD defaults for length or invalid length • Additional SyzMPF/z Reference Examples Guide is now available to better show how all of the commands work and how they can work together. ATTACH global parameter, to allow a default attachment of any JES sysout or other dataset to all generated email messages. This was requested to allow a default disclaimer to be attached to every email and/or to make sure that the condition JOB messages get sent along with them.
5.5	2/23/2017	Added ability to attach any Sequential file or PDS member via ATTACH DSN= PARSE2WORDS no longer needed for use of individual words by variable name for first 50 words (&W0 -&W49) of a message within a script. &VERSION command enhanced to show SyzMPF/z assembly date/time and version The following Variables have been added: &YYDDD - Julian Date &YYYYDDD - Julian date with 4 digit year &YYYY - 4 digit year &DDD - Julian day The MSGID and &MSGID can now be up to 48 bytes in length Attach JCL has been changed to ATTACH JESJCLIN to better show which JCL is being attached to the email

		<p><variable> \$ON\$ has been fixed, previously was set to always \$OFF\$</p> <p>For IF commands, and for all other areas, if the message being processed is a CONSOLE issued command (i.e. it was issued by an operator or MCS console instead of from a JOB or TASK), then the &TASKNAME variable is set to the "name" of the console that issued the message.</p> <p>Individual words of a message can be addressed by &Wn (where n is 0 through 49), previously this had to ALWAYS be a 2 digit number, now it can be &W1 or &W01 interchangeably.</p> <p>IF WORD nn has been enhanced to allow a single digit for the word number "nn" instead of always needing two digits. IF WORD 1 and IF WORD 01 are now equivalent.</p>
5.4	9/23/2016	<p>New &MSGTEXT variable to show the message text of the message being processed</p> <p>IF "ANYTHING" support added. This allows any value to be compared to any other value. Any variable can be compared to static text or other values.</p> <p>Added the Script name to the Script line in case of ABEND</p> <p>Minor updates to messages and decreased path length on some commands</p>
5.3	1/23/2015	<p>IF STARTED now supports Equal and NotEQ. Also supports stacking of commands to the same line via I (OR), + (AND), II (ORIF), ++ (ANDIF).</p> <p>IF STOPPED now supports Equal and NotEQ. Also supports stacking of commands to the same line via I (OR), + (AND), II (ORIF), ++ (ANDIF).</p> <p>IF STRING now supports Equal and NotEQ. Also supports stacking of commands to the same line via I (OR), + (AND), II (ORIF), ++ (ANDIF).</p> <p>IF COMMAND now supports Equal and NotEQ. Also supports stacking of commands to the same line via I (OR), + (AND), II (ORIF), ++ (ANDIF).</p> <p>IF TASKTYPE MSGTYPE now supports Equal and NotEQ. Also supports stacking of commands to the same line via I (OR), + (AND), II (ORIF), ++ (ANDIF).</p> <p>New Command IF ANYWords was created as an alias of IFWORDS because some users found it confusing to have IFWORD (specific word) and IFWORDS (any up to 4 words) so similar in name.</p>

		IFWORDS IFANYWords now supports Equal and NotEQ. Also supports stacking of commands to the same line via (OR), + (AND), (ORIF), ++ (ANDIF).
5.2	12/31/2014	<p>Added " " (orif) and "++" (andif) separator support for most "IF" related commands, (see individual commands for actual support). This support adds the ability to have two (or more) unrelated "IF" operations per command line.</p> <p>A new section of this manual was added, "IF-BASED COMMAND GENERAL INFORMATION", before the "IF-commands are defined, to show the use of the new comparators and separators</p> <p>The following commands now have support for "ANDIF", "++", as well as "ORIF" " ". The support is only limited to the length of the command line. :</p> <p>Example: If SYSID = TEST IF TASKNAME = CICS* (if the SystemID of the message originator is "TEST or the name of the task issuing the message was CICS* (generic)).</p> <p>IF SYSID IF ORIGIN IF MAXCC IF MSGID IF MSGTASK TASKNAME IF STARTED ACTIVE IF STOPPED INACTIVE IF DOW IF TIME CURRENTTIME IF LPAR IF SYSPLEX IF MAXRESULT IF SUBMETH IF MAXPROG IF CLASS IF MSGCLASS IF WORKLOAD IF SECGRoup RACFgroup IF NOTIFY IF USERID TASKOWNER IF OSVER IF OSREL IF OSMOD IF LVRC IF MSGTYPE</p> <p>Added IF TIME command (meant to replace the IFBEFORE/IFATER commands). Supports multiple options. Added &MAXRESULT variable Added &MAXPSTEP variable</p>

		<p>Added &TASKOWNER (same as &USERID) Added &SECGRoup (same as &RACFGRP) Removed LE and GE (less-than-or-equal-to, and greater-than-or-equal-to)parms from the IF LVRC command, customer suggestion as difficult to use. Added OR,AND,ANDIF,ORIF support. IFMSGTYPE command now supports and/or/andif/orif IFCOMMAND command now supports and/or/andif/orif IFSTARTED command now supports =,EQ,NE,^=.ORIF/ANDIF IFACTive is now an alias for IFSTARTED IFINACTive is now an alias for IFSTOPPED IFSTOPPED command now supports =,EQ,NE,^=.ORIF/ANDIF IF<variable> now supports generics and =,EQ,NE,^=.ORIF/ANDIF The following commands now support all of the following comparators: = EQ ^= NE > GT < LT / GE \ LE</p> <p>= and EQ EQUAL TO ^= and NE NOT EQUAL TO > And GT GREATER-THAN < and LT LESS-THAN / and GE GREATER-THAN or EQUAL TO \ and LE LESS-THAN OR EQUAL TO</p> <p>Commands: IF <variable> IF CLASS IF LVRC IF MAXCC IF MSGCLASS IF ORIGIN IF OSVER IF OSREL IF OSMOD IF STARTED ACTIVE IF STOPPED INACTIVE IF SECGRoup RACFGRP IF SYSID IF TIME CURRENTTIME IF NOTIFY IF USERID TASKOWNER IF WORKLOAD</p> <p>Eventually all commands will support all of the comparators, even though some of them will not really make sense to use for most sites, the support will be added solely for compatibility with all of the other commands.</p>
5.1	12/1/2014	Attachment support for JES files:

		<ul style="list-style-type: none"> - Any JES Dataset can be attached to an outgoing MaxCC email. <p>Attach=JESJCL - Attaches the JCL images as output from the converter</p> <p>Attach=JESMSGLOG - Attaches the JES message log (WTO's issued by the task)</p> <p>Attach=JESYSMSG - Attaches the system messages (JCL resolution messages and condition codes)</p> <p>Attach=JESJCLIN - Attaches the JCL as submitted or started. (JCL before JES processing)</p> <p>The attachments are included with the outgoing email, separated by generated separators with start and stop notification and number of lines transmitted.</p>
5.0	11/23/2014	<p>New Version (HTML support, QuickReplies, AND/OR support)</p> <ul style="list-style-type: none"> - Code Base for attachment support <p>The following commands now have support for "AND", "+", as well as "OR" " ". The support is only limited to the length of the command line. :</p> <p>Example: If SYSID = TEST PROD (if the SystemID of the message originator is "TEST or "PROD")</p> <p>IF SYSID IF ORIGIN IF MSGID IF MSGTASK IF TASKNAME IF DOW IF LPAR IF SYSPLEX IF SYSPLEXID IF MAXRESULT IF SUBMETH IF MAXPROG IF CLASS IF MSGCLASS IF WORKLOAD IF RACFgroup IF NOTIFY IF USERID IF OSVER IF OSREL IF OSMOD</p> <p>SyzMPF/z can now send both Text and HTML format email messages. This is particularly noticeable in the MaxCC and</p>

		<p>StepCC support since the information will be presented in a HTML table if HTML support is requested. The Startup keyword is "EFORMAT" and has options of "HTML", "TEXT" and "BOTH". "BOTH" denotes that both HTML and TEXT format is to be used in the same email, this is normally a good idea since some email clients cannot display HTML (or they may have the HTML email option turned off), so "BOTH" will send the messages in BOTH HTML and TEXT format within the same email package. EFORMAT is also available within scripts so that the format can be changed depending on requirements of the script at message processing time.</p> <p>MAXSTEPS command. MAXSTEPS is a START-UP and a script command. This will change the default number of per task (in the MaxCC email processing area) from the default of 25 steps to any number (up to 9999) of steps. Each step "slot" requires 100 bytes of storage to hold the message until it is "delivered" to SyzEmail/z for processing.</p> <p>Quick Replies. This allows for response to WTOR messages with no actual MPF script being necessary. This will allow complex replies to messages with less than 1% of the overhead of IBM's AutoWTOR processing or SyzMPF/z own WTOR processing. The MPFLSTnn member need only add an "AUTO(#xxxxxxxxxx)" entry to the line and SyzMPF/z will "reply" to the message with whatever is contained after the pound-sign (#), exactly as entered on the line.</p> <p>The code base has be radically changed so that the next update (currently in Beta testing) will be able to attach files to the Email when created. This support requires changes to SyzMPF/z, SyzCMD/z, SyzSPOOL/z and SyzEMAIL/z. The first implementation will allow any JES component of the JOB (JCL, MSGS, LOGS, or SYSOUT) to be attached to the email.</p>
4.2	4/10/2014	<p>Added several variables:</p> <p>&TO: (the current recipient (colon ':' required))</p> <p>&FROM: (the current sender)</p> <p>&CC: (the current copy address)</p> <p>&BCC: (the current Blind Copy address)</p> <p>&Subject (the current subject)</p> <p>New Startup parameters:</p> <p>SCRIPTMAX (maximum number of scripts allowed per message)</p> <p>--this is to provide support for GOTO/++INCLUDE/GOBACK and other new (and planned) commands so that an inadvertent loop condition will not be able to arise (default is 50))</p> <p>New script commands:</p> <p>::label (label name, object of GOTO)</p>

		<p>GOTO= (tells SyzMPF/z to search the current and all previous scripts used so far for the label presented)</p> <p>DEBUG= (If set to OFF, will not display Email created messages and some others)</p> <p>ENDMSG ENDMESSAGE (allows an email MSG: command to stop processing the message to do other script processes. Multiple MSG: commands can now exist for the email package and they will all be included in the order presented to the script).</p> <p>The following commands have a new possible parameter (NULL or NULLS. For instance for NOTIFY to compare for "no NOTIFY=" on the JOBCARD of the task.</p> <p style="padding-left: 40px;">IFNOTIFY = NULLS</p> <p style="padding-left: 80px;">or</p> <p style="padding-left: 40px;">IF NOTIFY = (NULL</p> <p>IFNOTIFY</p> <p>IFSYSID</p> <p>IFORIGIN</p> <p>IFMSGTASK IFTASKNAME</p> <p>IFLPAR</p> <p>IFSYSPLEX</p> <p>IFOSVER</p> <p>IFOSREL</p> <p>IFOSMOD</p> <p>IFDOW</p> <p>IFMAXRESULT</p> <p>IFMAXPROG</p> <p>IFCLASS</p> <p>IFMSGCLASS</p> <p>IFWORKLOAD</p> <p>IFRACFGRP</p> <p>IFUSERID</p> <p>IFSUBMETH</p> <p>Almost all IF commands now allow = ^= EQ NE comparators</p> <p>SIMULATE and NOSIMULATE commands added</p> <p>QUERY (allows the site to query SyzMPF/z to show statistics (QUERY DETAILS) of each message script's processing and the defaults (QUERY DEFAULTS) as currently set. This command can be added as an operator command (see the provided "@" and IEE305I members of the INSTLIB sent with the product)</p>
4.1	1/23/2014	<p>Major upgrade which adds the following Features:</p> <ul style="list-style-type: none"> • Email support – uses new SyzMAIL/z product • To:, From:, Cc:, Bcc:, Subject:, ReplyTo: Message support • Automatic End of task or during task condition code checking and email of those codes • Execution Class checking

		<ul style="list-style-type: none"> • MSGCLASS checking • Step and/or procstep result and program checking • Workload name checking • RACF group checking • NOTIFY ID checking • Submit method checking • Installation Defaults supported via Parmlib member • MessageID use statistics • Generic characters ('%' and '*') are now permitted for MSGTASK, TASKNAME, MSGID, LPAR, SYSID, and SYSPLEX • Debugging mode • Greater variable support &Maxcc, &Stepcc, &MaxSTEP name, &Max Program name (program name from highest condition code stepo) &RACF group, &Submit method, &execution class, &MSGCLASS, &WORKLOAD name, &Programmer Name, &NOTIFY ID, &Submit time and &Submit date of this task, &Start execution time and &Start execution date of this task • Maximum Message length of eMail • Variable support for SyzMail/z • Commands to do First (before script) • Commands to do LAST (After script) • SyzMPF/z defaults set in system parmllib member • The SCRIPT DSN is now a variable that can be set without JCL (it can be identified within the new parmllib member) • FIRST=/LAST= support to supply script members that will be executed before or after every script is processed. <p>Removed support in Version 4.0</p> <ul style="list-style-type: none"> • \$\$\$\$DFLT member no longer used (except via FIRST= parmllib parameter setting) • \$\$\$\$\$\$\$\$ member no longer used (except via LAST= parmllib parameter setting) • Hardware parameters no longer gathered for each script, now only gathered when necessary <p>Special attention should be taken with the new "Upgrade" section of this manual which covers the necessary items that must be addressed to upgrade from version 3 to version 4. This section will be updated over time to support upgrades from all versions of SyzMPF/z to whatever the current version(s) might be.</p>
3.3	2/11/2013	Minor message replacement for better readability
3.2	1/23/2013	<p>*****Because of customer feedback we have completely replaced the way variables are set, replaced and deleted. It was felt that the original method introduced with Version 3 and 3.1 was too difficult and cumbersome, so we have replaced the interface to persistent and non-persistent variables in this release as follows:</p>

		<p>so that that variable, when tested from any script (either SyzMPF/z or from the SyzCMD/z V8.0+ sister product) will show it's been removed that variable is no longer there. It was removed when CICS001 terminated.</p> <p>Support for deleting persistent and NON-persistent variables has been "enhanced" to make the process easier.</p> <p>Support added to test the success of the variable sub-command. The &LVRC (Variable Return Code) variable can be tested and displayed with new script commands.</p> <p>&LVRC display variable and insert variable added.</p> <p>IF LVRC script command added to allow for testing of the persistent and non-persistent variable commands.</p> <p>Persistent (and NON-persistent) variables can be created at any time in any script and checked from any other script, they do not have to be set in the script that they are being used in, which makes it a great way to pass long term parameters between scripts that may have nothing to do with each other except that they happen to be able to use information within those variables. NON-Persistent variables are just as useful, with the added benefit that when the task that they were attached to goes away, the variable goes away as well.</p>																												
3.0	11/23/2012	<p>New Support for persistent User variables, the variables can have any name the user desires (up to 12 characters in length) and the variable can contain any data (up to 16 characters in length) including spaces or any special characters the user might desire.</p> <p>New Support for "static" internal variables of the following type, all of the variables can be used in any command or can be written to the console (via WTO or WTOH command(s)):</p> <table><tr><td>&HH</td><td>2 character Hours</td></tr><tr><td>&MM</td><td>2 character minutes</td></tr><tr><td>&SS</td><td>2 character seconds</td></tr><tr><td>&HHMM</td><td>4 character Hours and minutes</td></tr><tr><td>&HHMMSS</td><td>6 character Hours and minutes and seconds</td></tr><tr><td>&MN</td><td>2 character Month (digits)</td></tr><tr><td>&DD</td><td>2 character Day (digits)</td></tr><tr><td>&YY</td><td>2 character Year (Digits)</td></tr><tr><td>&MMDD or &MNDD</td><td>4 character Month and day</td></tr><tr><td>&MNDDYY or \$MMDDYY</td><td>6 character Month, day and year</td></tr><tr><td>&MNDDYYYY or &MMDDYYYY</td><td>same with 4 character year</td></tr><tr><td>&DOW</td><td>3 character Day of Week (MON-SUN)</td></tr><tr><td>&MON</td><td>3 character month (JAN-DEC)</td></tr><tr><td>&LPAR</td><td>(up to) 8 Character LPARNAME</td></tr></table>	&HH	2 character Hours	&MM	2 character minutes	&SS	2 character seconds	&HHMM	4 character Hours and minutes	&HHMMSS	6 character Hours and minutes and seconds	&MN	2 character Month (digits)	&DD	2 character Day (digits)	&YY	2 character Year (Digits)	&MMDD or &MNDD	4 character Month and day	&MNDDYY or \$MMDDYY	6 character Month, day and year	&MNDDYYYY or &MMDDYYYY	same with 4 character year	&DOW	3 character Day of Week (MON-SUN)	&MON	3 character month (JAN-DEC)	&LPAR	(up to) 8 Character LPARNAME
&HH	2 character Hours																													
&MM	2 character minutes																													
&SS	2 character seconds																													
&HHMM	4 character Hours and minutes																													
&HHMMSS	6 character Hours and minutes and seconds																													
&MN	2 character Month (digits)																													
&DD	2 character Day (digits)																													
&YY	2 character Year (Digits)																													
&MMDD or &MNDD	4 character Month and day																													
&MNDDYY or \$MMDDYY	6 character Month, day and year																													
&MNDDYYYY or &MMDDYYYY	same with 4 character year																													
&DOW	3 character Day of Week (MON-SUN)																													
&MON	3 character month (JAN-DEC)																													
&LPAR	(up to) 8 Character LPARNAME																													

		<p> &SYSID (up to) 8 character SYSTEM NAME &CPUID CPU serial number &CPUTYPE CPU type (device name) &SYSPLEX (up to) 8 character SYSPLEX name &SYSPLEXID 2 character SYSPLEX ID &TASKNAME Name of the task that issued the message &TASKID 8 character tasks ID (i.e. JOBnnnnn) &TASKTYPE 3 character (JOB, STC, TSU) &ORIGIN Command or message system of origin &ASID Address Space ID &SCRIPTID name of the script member being processed &MSGID (up to) 48 character message ID we are on &CONSOLE (up to) 8 character CONSOLE ID that issued msg &CONID CONSOLE ID the issued message </p> <p> New Support for IF<variable> so that the user can do (or not do) work based on the settings of the user variables that may be specified. The user can compare for the mere existence of the variable or the actual contents of the variable named in the command. </p> <p> New ESTAEX support was added to handle any ABEND the may happen even if it is outside the SyzMPF/z code, (for instance within MPF itself). </p> <p> Ability to test whether a message or command was issued from a program or from an operator console and who issued that command if from a subsystem (i.e. issued as a command from SDSF by a TSO user). This allows the site to generate its own operator commands, since SyzMPF/z will be able to differentiate between a "command" that was issued and a simple WTO or WTOR. </p> <p> Now any number of blanks between most command pieces to make the SyzMPF/z commands more free-form. </p>
2.4	7/31/2012	<p> Changed to allow UPPER or lower case variable names. Previously they had to all be in UPPER case. </p> <p> Make PARSE2WORDS command routine ECHO to the console if ECHO settings are correct. </p> <p> Corrected redisplay of corrected messages at end of message processing </p> <p> Explained the NOCONSOLE non-command. This is how you can have a message still go to SYSLOG and the JOBLOG, but not go to the Operator's console. </p>
2.3	2/15/2012	<p> Replaced WTOE routine because some sites experienced too many messages. </p> <p> Added support for MLWTO suppression while still allowing the message to be processed logically. (Client Request) </p>

2.2	1/12/2012	Minor Update to correct some small logic issues with message processing of multiple action messages
2.1.0	11/23/2011	Minor Updates to correct spelling on several messages
2.0.0	9/22/2011	<p>Re-engineered the structure of the base module and sub-modules so that there are no longer any dependencies on message size of capture buffers.</p> <ul style="list-style-type: none"> - Conversion of MGCR to MGCR_e - Variable length DSA getmains/freemains - New Simulation Mode allows for live testing without actually having commands be issued - New expanded ESTAE for Sub-routine support - Streamlined WTO support. <p>MODE=SIMULTE processing now available MODE=QUIET processing now available GLOBAL command processing via \$\$\$\$DFLT script member is now available. PARSE2WORDS</p> <ul style="list-style-type: none"> - script command added to allow the site to "parse" the MPF message into all of the individual words (up to 50) that make up the message. <p>&Wnn</p> <ul style="list-style-type: none"> - Allows the use of any (or all) of the individual words of the MPF message being processed to be used in ANY command to be issued. This is an enhancement to the existing "GET WORD" script command. <p>ECHO=PARSE</p> <ul style="list-style-type: none"> - added to allow site to display the individual words of a message in response to a PARSE2WORDS command <p>GET WORD no longer requires any column alignment of any parameters, previously 1 space between parms was forced GET STRING no longer requires any column alignment of any parameters, previously 1 space between parms was forced IF WORD no longer requires any column alignment of any parameters, previously 1 space between parms was forced IF STRING no longer requires any column alignment of any parameters, previously 1 space between parms was forced</p> <p>Repaired several bugs located in Alpha Testing</p> <p>Released to Beta test (9/22/2011)</p>
1.5.0	3/1/2011	Redesign of EstaeX recovery code to show exact areas of MPF problems and IBM MFA issues. Based on known problems with IBM Message Flood Automation, we have redesigned the EstaeX code to snap more information when a problem is found. This will also reduce the possibility that IBM's MFA will cause a recursive abend in MPF exits should it replace the entry point of our module.
1.4.0	1/23/2011	Corrected small error in IFAFTER processing. Corrected spelling errors in various messages.
1.2.0	11/23/2010	Added logic to dis-allow SKIPTO branch to the same member we were currently using, this can result in a LOOP condition under most circumstances.

		Minor updates to the spelling on some messages. Removed duplicate messages.
1.1.1	10/3/2010	GA release Added SKIPTO command to allow MPF command script chaining in case user wants to use more than 100 lines of command scripting, allows unlimited number of chains. Various minor fixes to messages.
1.0.g	9/29/2010	Allow full subset of system symbols in any command Fixed error in cross sysplex command dubbing on slow connections
1.0.f	9/20/2010	Allow use of system symbols in the commands that are issued (not in WTOs) Start DUMP&SYSNAME results in Start DUMPPROD Fixed documentation error changing HIGHLIGHT to HIGHLITE Added new command LOWLIGHT (this one is spelled right) Added new MSGCOLOR sub-command of LOWLIGHT Added new command IFORIGIN for origin system of message if on plex Changed ECHO command on OFF or NONE to not echo that "ECHO" command Changed REPLY command to correct QUIET mode processing.
1.0.e	9/08/2010	Changed SyzMPF/z program to (re) allow 'n' for a REPLY command, due to popular request.
1.0	7/12/2010	Version 1 Beta Program Release

Table of Contents

Contents

Revision History	ii
<i>Table of Contents</i>	17
1. Introduction	22
Purpose	23
Additional Examples	24
Related Documents	24
Conventions	24
Problem Reporting Instructions	25
2. Overview	26
Functional Description & figure view	29
3. Installation Instructions	30
Standard installation Procedures	30
STEP 1:	30
STEP 2:	30
STEP 3:	30
STEP 4:	31
STEP 5:	31
STEP6:	33
STEP7:	34
STEP8 (highly suggested):.....	35
STEP 9 (optional):	36
STEP 10:	37
STEP11:	37
Required steps to process console ALL message traffic.	39
Some basic things to know:.....	41
Helpful Syntax rules for MPFLSTxx	43
Controlling message management	44
Special MPFLSTxx AUTO actions which can be taken with SyzMPF/z	45
QuickReply with MPFLSTxx AUTO(#xxxx...) actions which can be taken with SyzMPF/z	46
4. Global Settings (system parmlib)	48
Sample SYZSUB00 “SYSTEM PARMLIB” DEFAULTS member.....	49

MPFDSN.....	52
ECSAMAX	54
ECHO.....	55
DEBUG.....	59
(email related) EFORMAT EMFormat.....	60
(email related) SYZMAIL.....	61
(email related) MAXMSG	62
(email related) MAXSTEPS	63
SCRIPTMAX.....	64
FIRST.....	65
LAST.....	66
(email related) To:.....	67
(email related) From:	68
(email related) Replyto:.....	69
(email related) Cc:.....	70
(email related) Bcc:.....	71
(email related) Subject: Subj:.....	72
(email related) TimeZone or TZ =	73
(email related) ATTACH	74
Mode Commands:.....	77
MODE=SIMulate NOSIMulate	78
MODE=QUIET NOQUIet	79
STATistics Stats.....	80
VERBOSE	81
SIMULATE	82
TRACE	83
5. Automatic End of Task Email.....	84
Sample script for Error processing.....	85
IEFC452I Script	85
Sample script for regular task processing.....	86
MAXCC script for \$HASP395 & \$HASP396	87
6. Control Commands:.....	89
* (Comment)	89
&REPLYID.....	90
&STR (or &STRING).....	91
&variables - STATIC Variables.....	92
TIME Related STATIC variables:	93
DATE Related STATIC variables:	94
&ASID - Address space ID STATIC variable:.....	95
&BCC: - Email Blind Carbon Copy (“:” is required) STATIC variable:	96
&CC: - Email Carbon Copy (“:” is required) STATIC variable:	97
&CONID - Console ID STATIC variable:	98
&CONSOLE - Console name STATIC variable:	99
&CPUID - CPU Serial Number STATIC variable:	100
&CPUTYPE - Mainframe Model Type STATIC variable:	101
&DOW - Day of Week STATIC variable:	102
&EXECCLASS- Execution CLASS of this task	103
&From: - Email From (“:” is required) STATIC variable:	104

&LPAR - Logical Partition Name STATIC variable:	105
&MaxCC - Maximum Condition code (so far):.....	106
&MAXPROGRAM- Step Program with the Maximum condition code (so far):.....	107
&MAXPSTEP- ProcStep Name with the Maximum condition code (so far):.....	108
&MAXRESULT- Highest Task Result	109
&MAXSTEP- Step with the Maximum condition code (so far):.....	110
&MON - Month STATIC variable:	111
&MSGCLASS- MSGCLASS of this task:	112
&MSGID - Current Message being processed STATIC variable:.....	113
&MSGTEXT - Current Message being processed - STATIC variable:	114
&NOTIFY- JOB card or /* JECL NOTIFY= :	115
&ORIGIN - Origin of this task (System name) STATIC variable:.....	116
&PGMRNAME- JOBCARD Programmer Name of this task:	117
&SECGRoup &RACFGRP - Security group of this task:	118
&SCRIPTID - Currently executing Script name STATIC variable:.....	119
&SYSID - System ID STATIC variable:	120
&SYSPLEX - SysPLEX Name STATIC variable:.....	121
&SYSPLEXID - SysPLEX ID STATIC variable:.....	122
&STARTDATE- Date this task began execution	123
&STARTTIME - Time this task began execution	124
&SUBDATE- Date this task entered (or submitted to) the system:	125
&SUBJECT: - Email SUBJECT (“:” is required) STATIC variable:.....	126
&SUBMETH - Submit Method (resource) of this task:	127
&SUBTIME- Time this task entered (or submitted to) the system:	128
&TASKNAME - Task Name STATIC variable:.....	129
&TASKID - TaskID (TASK-number) ID STATIC variable:.....	130
&TASKTYPE - Task Type STATIC variable:.....	131
&TO: - Email TO id (“:” is required) STATIC variable:	132
&USERID &TASKOWNER - Task Owner, submitter or USER= of this job :	133
&VERSION - Display the SyzMPF/z version, assembly date and time	134
&<variable> - use previously set user variable:	135
&WORKLOAD- WLM WORKLOAD name of this task:.....	136
DECVAR or DECVART	137
nn	137
INCVAR or INCVART	138
nn	138
SETVAR - Setting Persistent Variables	139
SETVART - Setting NON-Persistent (temporary) Variables.....	142
REPVAR - Replacing Persistent Variables.....	146
REPVART - Replacing NON-Persistent Variables	149
DELVAR – Delete any user Variable (Both NON-Persistent & Persistent).....	153
&LVRC Display the Last Variable Return Code.....	155
&Wnn &Wn a specific word of message being processed	157
&WORD	158
::labelname	159
AMRF ON/OFF	160
ASIS.....	161
AUTOMATE	162
AUTOOFF	163
DEBUG	164
ECHO.....	165

ELSE	168
EMAIL and SMS/Text related Commands.....	169
Email:	170
EndMessage or EndMsg	171
SENDMAIL	173
To:	175
From:	177
Cc:	179
Bcc: Blind carbon copy	181
ReplyTo:	183
SUBJect: or Subj:	185
Message: or Msg:	187
MAXMSG =	189
ATTACH =	190
ENDIF	192
(EXIT)	193
GET REPLY	194
GET STRing	195
GET WORD	197
GOTO	200
HIGHLITE	201
“IF” Based Command general Information:	202
IF “ANYTHING” Compare any variable, text, string, etc	207
IF <variable> (Generics Supported)	208
IF ACTive IF STARTed (Generics Supported)	210
IF AFTER (Replaced by IF TIME, phasing command out)	211
IF BEFORE (Replaced by IF TIME, being phased out)	212
IF CLASS (Generics Supported)	213
IF COMMAND	214
IF CURRENTTIME IF TIME (Generics Supported)	216
IF DOW (Generics Supported)	217
IF INACTive IF IFSTOPped (Generics Supported)	219
IF LPAR (Generics Supported)	220
IF LVRC (Generics Supported)	222
IF MaxCC (Generics Supported)	224
IF MAXRESULT (Generics Supported)	226
IF MAXPROGRAM (Generics Supported)	227
IF MAXPSTEP	228
IF MAXSTEP	229
IF MSGCLASS (Generics Supported)	230
IF MSGID (Generics Supported)	231
IF MSGTASK TASKNAME (Generics Supported)	233
IF MSGTYPE	235
IF NOTIFY (Generics Supported)	236
IF OFFLINE	237
IF ONLINE	238
IF ORIGIN (Generics Supported)	239
IF OSMOD (Generics Supported)	240
IF OSREL (Generics Supported)	241
IF OSVER (Generics Supported)	242

IF RACFgrp IF SECGRoup (Generics Supported).....	243
IF SECGRoup IF RACFGRP (Generics Supported)	244
IF StepCC(Stepname) or (Stepname.Procstep)	245
IF STARTED IF ACTive (Generics Supported).....	246
IF STOPPED IF INACTive (Generics Supported).....	247
IF STRING.....	248
IF SUBMETHod (Generics Supported).....	250
IF SYSID (Generics Supported).....	251
IF SYSPLEX (Generics Supported)	253
TASKOWNER IF USERID (Generics Supported)	254
IF TIME IF CURRENTTIME (Generics Supported).....	255
IF USERID TASKOWNER (Generics Supported)	256
IF WORD (Generics Supported).....	257
IF WORDS.....	259
IF WORKLOAD (Generics Supported)	260
LOWLIGHT	261
MSGCOLOR	262
NOCONSOLE	263
NOJOBLOG	264
NOSYSLOG	265
PARSE2words	266
PARSEDELIMITER.....	267
QUERY	268
REPLY	271
SKIPTO.....	272
STRIP.....	273
SUPPRESS	274
TimeZone or TZ =.....	275
SYSMAIL =.....	276
WTO	277
WTOH.....	278

[The rest of this page has been intentionally left blank]

1. Introduction

SyzMPF/z is part of the Syzygy Automation suite of products. The Syzygy Automation Suite provides z/OS sites with a complete and comprehensive capability to automate the entire processor complex. The Suite is made up of several main products;

SyzAUTO/z, the command and task scheduling facility, which allows the z/OS site to schedule any system or subsystem command, submit batch JOBS, start tasks and control timed operations on a 24x7 basis.

SyzCMD/z, the Command Scripting facility, provides the z/OS site with a comprehensive means to start automated scripts to perform automated and complex functions with complete nested IF/THEN/ELSE logic, allowing the site to control all simple and complex operations in an orderly and efficient manner.

SyzSPOOL/z, the JES SPOOL maintenance facility, which offloads spool data from JES2 or JES3 to separately addressable data sets, and allows users to access the offloaded output interactively from TSO/ISPF or via the Web from any standard web browser. Full security is maintained for access to the offloaded spool data, which is able to be controlled via any storage manager (HSM, ABR), or via SyzSPOOL's own internal dataset allocation control methods. The Data may be viewed, sent via FTP or via Email in any of several output formats including PDF, WORD, HTML, XML and several others.

SyzMPF/z, the Console Message Processing Facility, allows the z/OS site to respond to any Console-type message with predefined scripts allowing complete interactive control of any console situation or message. SyzMPF/z provides an automated means for answering any request or responding to any console based event, (job end, start, abend, message from any job or task, etc.). In short, any console event or message can be handled by the facilities of SyzMPF/z.

SyzMAIL/z (add-on), the end of task notification facility. SyzMail/z captures the maximum condition code, plus all (or any) of the Step condition codes, plus other task related information (execution time, CPU time used, I/O's etc.) and sends them to any Email address or destination. User's no longer have to logon to the system to see if a task completed and what it's condition codes were, they can be notified via email, which can be encrypted and delivered directly from the mainframe.

SyzMail/z, the system eMail facility. SyzMail/z supports the monitoring and gathering of eMail messages from all Syzygy products and Syzygy supported Customer API's, and some other vendor products. This facility allows the use of Nicknames for the To:, Cc:, and Bcc: commands, so that multiple recipients can be supported with a single nickname entry. The facility supports storage of the email messages until they are successfully passed off to the sites SMTP server for distribution. Full statistics are provided for all functions.

SyzText/z, the system SMS Text and short eMail facility. SyzText/z supports the creation of short (up to 100 byte) SMS text or Email message from the system console or via a batch job or step. This facility allows the use of Nicknames for the To:, Cc:, and Bcc: commands, so that multiple recipients can be supported with a single nickname entry.

Purpose

The products that make up the Syzygy Automation Suite are designed to offload the burden of supporting the normal day to day operation of the computer center. The facilities allow the site to spend their time and money on things that really matter, instead of maintaining the same old processes which can be automated by the components of the Automation Suite. The site will conserve manpower, and resources.

Additional Examples

This 2017 Reference adds more Examples to this guide and was developed in order to better show how SyzMPF/z commands and capabilities work, not only within this product, but how to use this product in conjunction with other Syzygy Automation products to more effectively monitor and control the client site's z/OS operating system and subsystems. It is the intention of Syzygy Client Services that this manual contain not only detailed instructions on how to issue the commands and write the scripts, but also why they work and how to approach actual production environments in order to implement z/OS system automation in a simple and straightforward way. If at any time you don't understand how a specific command or script works, how to approach automation tasks, or if you have examples or explanations that you feel other's might benefit from within this manual, please send your request or comment to:

RefGuide@SyzygyInc.com

Related Documents

SyzAUTO/z	Installation and Operations Guide
SyzCMD/z	Installation and Operations Guide
SyzSPOOL/z	Installation and Operations Guide
SyzMPF/z	Installation and Operations Guide (this guide)
SyzMail/z	Installation and Operations Guide
SyzTEXT/z	Installation and Operations Guide

Conventions

Where present, z/OS, OS/390 and MVS may be used interchangeably. It is not meant that these products are exactly the same, but they are sufficiently alike that for the purposes of this manual, they can be thought of as the same family with similar or identical support constraints with respect to the products covered in this manual.

Problem Reporting Instructions

Problems should be reported to:

ClientSupport@SyzygyInc.com

Problems with Beta versions of this product should be reported to:

Beta@SyzygyInc.com

2. Overview

The SyzMPF/z Message processing utility provides a z/OS data center with the capability to automatically respond to any and all sysplex wide system console or hardcopy messages by executing simple or complex “scripts” of MVS, JES2, JES3, CICS, DB/2, Unix System Services (USS) or any other subsystem or “on the spot” generated commands.

In conjunction with the SyzMAIL/z product add-on, SyzMPF/z can send email or SMS text messages of any length to any number of users. This capability is especially helpful when it is desirable to send the condition codes from a task. This can be accomplished at any point in the task, including the end of task.

The SyzMPF/z command scripts must be kept in any “standard” PDS format data set, PDS/e is currently not supported due to lack of support for PDS/e datasets before the IBM supplied PDS/e “helper” space is started. The members of this PDS are referred to by the name “scripts” and are loaded and reloaded on demand, or automatically during the z/OS system IPL process. The SyzMPF/z product functions in two modes dependent upon whether or not the site has elected to run the authorized subsystem (SyzygySS, which is the “suggested” method of operation). If you have elected to run the subsystem, then SyzMPF/z is implemented at IPL time and any messages generated after the subsystem is loaded (during initial IPL when the IBM parmlib IEFSSNxx member is processed). This means that “most” of the system level sub-systems are not yet initiated and can be responded to by SyzMPF/z, such as starting JES, and making system configuration changes before the spool subsystem is activated. If the site elects NOT to use the SyzygySS subsystem, then messages cannot be responded to until after the MPF memory areas are initialized via the standard startup started task or Batch JOB. The function of the product is the same in either case, it's just that sites who wish to automate IPL-time messages will normally want to activate the SyzygySS code during IPL.

While not quite as flexible as the companion product SyzCMD/z, due to the fact that SyzMPF/z, as an MPF message processing component, must run under IBM's MPF

processing rules and therefore cannot “wait” for things to happen and can only “react” to what is actually happening. This means that SyzMPF/z can more rightly be termed a “CONSOLE AUTOMATION” product, because all messages (whether or not they are actually sent to a “real” console, (they can be JOBLLOG or SYSLOG messages which do not display on the console). That still covers a LOT of automation ground that SyzMPF/z can be used to automate, but limits the user to NOT (directly) using script facilities that result in a system wait. What this means for writing your SyzMPF/z scripts is that while you can check on events like “IF a TASK is running”, you CANNOT wait for that task to start up as you could with SyzCMD/z. However, SyzMPF/z can use/attach SyzCMD/z facilities to perform those types of functions where “waiting around” might be advantageous. SyzMPF/z can still automate the operating system IPL, Shutdown, and almost any operator function in between based on messages that are issued or by responding to Operator Commands, it’s just not as good a fit for some complex functions as the SyzCMD/z product might be. Conversely SyzCMD/z is not a good fit for CONSOLE AUTOMATION because it’s not built to “watch” the message traffic like SyzMPF/z does. Command scripts can be built to accomplish extremely complex tasks and assigned any command name the site may decide to use. The possibilities are extremely exciting.

Remember, SyzMPF/z is not simply a “dumb” scripting facility for issuing commands based on console message, it has a direct interface with many internal system functions and provides many special features for determining what needs to be done and when. SyzMPF/z also (with the addition of the SyzMAIL/z add-on product), is able to send email or SMS text to any address or phone, to notify them that something has happened, and attach detailed information and even SYSOUT that may be of interest to the recipient of the message. SyzMPF/z provides a very simple scripting language that allows the site to harness extremely powerful command logic, including nested IF/THEN/ELSE logic as well as variable substitution and variable checking as “WORD”s of the message or as “Strings” within the message.

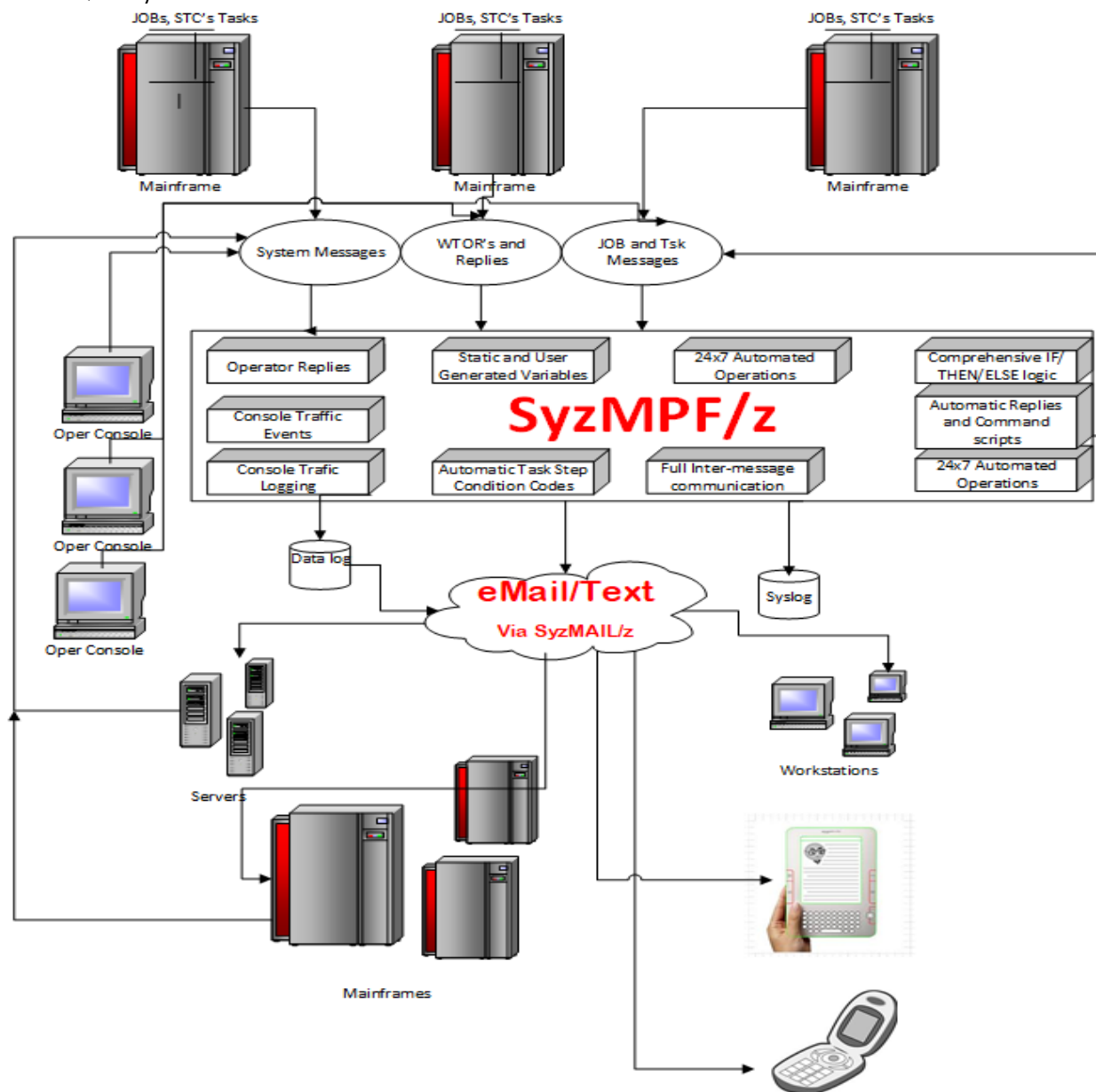


Full support of MLWTO’s (Multi Line WTO’s) are supported on every message, they are treated as a single concatenated line (although quite possibly a very long line of up to 255 characters) by SyzMPF/z.

Facilities are provided to allow the site to perform script/command logic only IF some task is active, or not active, or only if (or if not) a certain day of the week, or time of day. The site can “count” messages and only do something when they happen more than “x” number of times within a certain time frame, and can even write to a “runtime log” and ship that log to someone via email or SMS text (assuming the SyzMAIL/z add-on is acquired). Using the SyzMAIL/z add-on, SyzMPF/z can automatically send the condition codes and other step information at task end (or any time while the task is executing), to any recipient via email or sms text utilizing a great deal of logical possibilities. This means that jobs which execute and get a condition code above some preset value, can be set so that they will notify the person responsible, based on any of almost 50 available factors such as, jobname, programmer name, owner, RACF group, time of day, day of week, class, msgclass, tapes used, messages generated during processing, and many more. This is all possible with no changes necessary to any JCL or task/job in the system.

Functional Description & figure view

SyzMPF/z is implemented as a very powerful MPF exit via entries in the sites MPFLSTnn member of the system parmlib concatenation. SyzMPF/z is composed of several 31bit and 64bit subroutines (CSECTS) generated into a single load module, which are executed from an "APF authorized" library in the system Linklist. Many of the functions of SyzMPF/z can be protected by calls to the sites Security system (RACF, ACF/2 Top Secret, etc).



3. Installation Instructions

Standard installation Procedures

STEP 1:

The software is sent to the authorized site representative's email address. It will be attached to this E-Mail communication as a .ZIP file. You should save or copy this file to a suitable directory on your PC.

STEP 2:

Use PKZIP or some other ZIP/UNZIP program to unzip the file to the same or another PC directory. You will thereby obtain an EBCDIC format file with the following name structure:

software.XML where "software" is the name of the name of the Licensed Software Product that you have ordered.

STEP 3:

PRE-ALLOCATE THE RECEIVING XMIT FILE(S) ON YOUR MAINFRAME

Pre-allocate a container file to hold the product's load module library transmit format library n MVS (e.g. 'userid.software.**linklib**.dateofshipment.xmit') with the following DCB.

Space: 1 Cylinder
Organization : PS
Record format : FB
Record length : 80
Block size : 3120

Pre-allocate a container file to hold the Installation Source/samples transmit format library on MVS (e.g. 'userid.software.**instlib**.dateofshipment.xmit') with the following DCB.

Space: 1 Cylinder
Organization : PS
Record format : FB
Record length : 80
Block size : 3120

STEP 4:

UPLOAD THE software.XMI FILE(s) FROM YOUR PC TO YOUR MAINFRAME.

Using FTP or the file transfer component of your TCP/IP telnet emulator, specifying BINARY transfer, upload the "software.XMI" files to your mainframe. You should pre-allocate the destination dataset as outlined in STEP 3 because some mainframe site configurations do not automatically allocate the proper format container for the uploaded XMIT file.



BE SURE TO USE BINARY FILE TRANSFER, DO NOT USE ASCII TRANSLATION.

STEP 5:

RECEIVE THE XMIT FILE TO CREATE THE LINKLIB and INSTLIB PDS(s)

On your MAINFRAME perform a RECEIVE command on the uploaded XMIT files from step 4 above as follows:

NOTE: If operating under ISPF use panel option 6, or you should exit ISPF to TSO READY state.

COMMAND:

```
RECEIVE INDATASET('userid.software.LINKLIB.date.xmit')
```

***where 'userid.software.LINKLIB.date.xmit' is the pre-allocated file in STEP 3.

After completing the above command you will be prompted with something similar to the following:

```
INMR901I Dataset DATA.SET.NAME from userid on ?????????
INMR906A Enter restore parameters or 'DELETE' or 'END' +
```

at this prompt you can name the dataset something other than the "default" name that the file was shipped as, which may not meet your site's Dataset Naming requirements.

You can enter:

```
DSNAME('userid.software.linklib')
```

... where "userid.software.loadlib" will be a new load module PDS that you have selected which will contain all of the modules required for the correct operation of the licensed software that you have ordered. Do not pre-allocate this output dataset name.

NOTE: PLEASE BE SURE to choose a dataset name that does not already exist and which will be RACF ACCESSABLE by YOU. We suggest that you use your TSO userid as the HLQ (High Level Qualifier) to assure RACF permission.



*Note: Some mainframe sites will default to a PUBLIC volume if DF/SMS is not set up properly. If your site falls into this category, you might want to use the following format of the response to keep your dataset from being allocated to your WORK volumes and being scratched before you are ready:

```
DA('userid.software.linklib') VOL(volser) UNIT(unitname)
```

*** where "volser" is a DASD volume at your site (e.g. TSO001) and "unitname" is the esoteric unitname which governs that volser (e.g. SYSALLDA or 3390).

Repeat this step (STEP 5) specifying the name of the (Installation Library) INSTLIB dataset:

```
RECEIVE INDATASET('userid.software.INSTLIB.date.xmit')
```

***where 'userid.software.INSTLIB.date.xmit' is the pre-allocated file in STEP 3.

After doing the above command you will be prompted with something similar to the following:

```
INMR901I Dataset DATA.SET.NAME from userid on ????????  
INMR906A Enter restore parameters or 'DELETE' or 'END' +
```

at this prompt you can enter:

```
DSNAME('userid.software.INSTLIB')
```

... where "userid.software.INSTLIB" will be a new Source/Samples PDS that you have selected which will contain all of the Installation related samples required to help with the correct operation of the licensed software that you have ordered. Do not pre-allocate this output dataset name.

STEP6:

***If a beta version is installed you MUST REPLACE THAT 30-DAY VERSION OF YOUR LICENSED SOFTWARE

You must copy the load module(s) shipped and received by you in the previous steps to the library that currently contains your 30-day trial version of the licensed software. It is strongly suggested that you place the provided module(s) in a system LinkList library. You do not have to use the same library, but you will need to delete the old 30-day trial version of the software if you do not use the same library or else the software will cease to function when the 30-day trial period ends. In some cases the load module(s) shipped will need to be placed into an "APF AUTHORIZED" library. If APF authorization is required it will be noted in the identification section(s) above. Normally it is always safe to copy the load module(s) to the same library as your 30-day trial versions so that no system or user JCL members will need to be updated.

STEP7:**Add the SYZSUBnn parameter to your System Parmlib Concatenation.**

In order to support greater control over the capabilities of SyzMPF/z, a new feature was added to version 4.0 of the product which allowed the site to have complete control over all operations of the SyzMPF/z product. This member, (which can exist in ANY level of the SYSTEM PARMLIB concatenation), will hold all of the "defaults" and "settings" for global SyzMPF/z operational capabilities. If this member does not exist, then SyzMPF/z operates the same as pre-version 4 facilities and in some cases will not be able to provide some capabilities of the product. In addition, if not "startup" member is found, some features may not work correctly. This "startup" facility is named syzSUBnn (where nn is normally "SS") because it's controlled by the Syzygy Subsystem and NOT the SyzMPF/z product itself. It was determined that at some later date, we may use a SyzMPFnn member for even more SyzMPF/z "detailed" control, but that has not been necessary (so far).



*****NOTE: if the parmliib defaults member is not found, some features of SyzMPF/z may not function or may function differently.**

The default member name in your parmliib concatenation is SYZSUB00 (zero-zero). You can override that default at run time, (see JCL PARM option later in this manual), and there are several supported parameters which are outlined in the PARMLIB Startup Member Parameters section below. Any information which is not understood to be a valid parameter is ignored and a corresponding message is generated to the z/OS console. If you wish to have comments (and we suggest you do) within that member, you "should" use an "***" (asterisk) in column 1 of the line, and while it is not absolutely necessary (because any unrecognized command is ignored), it would be a good idea because at some future time the restrictions on "invalid" parameters might be implemented and it will cut down on the error messages at startup because comments are not displayed.

STEP8 (highly suggested):

Decide whether to install the Syzygy Subsystem.

The SyzMPF/z product is made up of two parts, one part processes the MPF scripts and places them in a special area of memory (within the ECSA (Extended CSA)), so that access to the scripts can be obtained by the SYZMPF/z utility in a direct memory access mode, which totally eliminates any physical I/O and system WAITs.

IBM *strongly* suggests that when dealing with console messages via the z/OS integrated IBM MPF (Message Processing Facility) that any process which executes a WAIT or performs any action which results in a WAIT (like opening a file to read the scripts), be "completely" eliminated. With the high number of messages that are processed by IBM's MPF, it would be a heavy drain on the system to cause IBM's MPF to wait for any reason. As such, we decided to load the SyzMPF/z scripts into a memory location so that no MPF WAITs related to opening and closing or fetching the scripts will EVER occur.

The facility that loads the SyzMPF/z scripts into the ECSA area is called SyzygySS. It can be run as a started task (or batch JOB) at any time to load, update or delete the existing scripts in memory (a sample is provided in the included INSTLIB). Additionally, it can be automatically executed at IPL time as an authorized SUBSYSTEM, via a small update to the IEFSSNxx member (a sample provided in the included INSTLIB) of "SYS1.PARMLIB". The advantage to using the authorized subsystem method of operation is that as soon as the subsystem has been activated and the IBM MPF facility is made active, which happens during the initial IPL process of z/OS, SyzMPF/z can begin to process the message traffic. If you elect not to use the authorized subsystem method, you cannot process any MPF message traffic until you run the SyzygySS program via a started task or batch JOB. Normally that means that you will have to wait until JES is started first, (or you could run the Started task with the "SUB=MSTR" parm to run it without JES being active).

For testing purposes, it's suggested to just execute SyzygySS as a started task (delivered as a sample procedure called SYZMPFZ), and then when you are satisfied with the operation of SyzMPF/z, you can update the IEFSSNxx member of parmlib.

Any time you want to change any or all of the scripts, you would only need to start the SYZMPFZ started task and it will remove and reload all of the scripts from the script library just as happens during IPL if you have implemented the SyzygySS entry in the parmlib IEFSSNxx member.

Setting up the IEFSSNxx parmlib member to support the SyzygySS authorized Subsystem.

Add the following two lines to your IEFSSNx member in your PARMLIB concatenation, you can add them anywhere before or after any entries you might already have:

**** as of version 4, the INITPARM entry above (INITPARM('DSN=SYS1.MPF.SCRIPTS')) is no longer necessary unless the site decide NOT to use the parmlib SYZSUBxx member.

The above parameters are defined as follows:

SUBNAME(**SYZ**) : The subsystem name, "SYZ" is required

INITRTN(**SYZSUBSS**) : The program name to start the subsystem
(This program needs to be in LINKLIST)

INITPARM(**'DSN=SYS1.MPF.SCRIPTS'**) (**this is not necessary as of version 4 or above**) The data set name that holds your SyzMPF/z scripts.
(Any name is acceptable, it must be cataloged in the master catalog)

OR
INITPARM('ID=xx') : Sets the system parmlib member to something OTHER than
SYZSUB00 i.e. ID=99 is SYZSUB99

You can activate the newly created IEFSSNxx member without an IPL by using the SETSSI command:

```
SETSSI ADD,SUBNAME=SYZ,INITRTN=SYZSUBSS,INITPARM='DSN=SYS1.MPF.SCRIPTS'
```

STEP 10:

Create the MPF command scripting PDS data set.

Create a standard PDS data set:

DSORG	PO
RECFM	FB
BLKSIZE	any ("9440" is a good choice)
Primary	10 CYL
Secondary	10 CYL
DIR Blks	50

STEP 11:

Place your sample SyzMPF/z scripts into the PDS you created above. This step can be performed with any utility (IEBCOPY) or via ISPF option 3.3, or any other method that is acceptable to your site to copy the sample scripts into the library.

The message processing scripts are created from the SCRIPT DSN (identified previously), and are loaded as a “package” into protected ECSA storage. The “default” total size of this area is 1MB, but can be altered by the “ECSAMAX=nnnK|M” startup parameter. The size will fluctuate dependent upon how many MPFLSTnn messages are to be processed and the length of the scripts used to process those messages, but the total size normally will not exceed 1Megabyte for even the largest sites. SyzMPF/z tells you at startup how much space was allocated, how much was actually used and how much is “wasted”, you can alter the ECSAMAX= parameter to get a good fit. If the size selected is not large enough to hold all of the scripts, SyzMPF/z will tell you about it (highlighted error), and will load as many scripts as possible until it runs out of space. Even though your scripts might contain “comments”, they (the comments) are not loaded into memory so commenting your scripts is beneficial and won’t use any extra space.

The “average” size for most sites is between 120K to 300K. This area is used by SyzMPF/z alone, and need not be storage protected in any way. The area will be identified by size and

location when it is created, and can also be easily located by the IEANT token name of “SyzygyMPFScripts”.

This area is normally loaded via a Started task, and JCL is shipped for that started task in the “Samples” dataset. The site may elect to perform this function via a batch job, but it is “normally” performed at SYSTEM IPL time via an entry in IEFSSNxx, but the provided JCL will allow the scripts to be reloaded in the event that they are changed or altered. The IPL load can be accomplished via the IEFSSNxx entry identified in step “9” above, but can be performed at any other time. Note that message processing cannot begin until the ECSA are loaded, so it makes most sense to do this automatically at IPL time.

There are two parameters possible for the JCL, “ID=xx” and “DSN=SCRIPT.Data.set.name”, neither is required, but “ID=xx” is suggested, unless you only wish to use the SYZSUB00 (zero) member, as it’s the default..

The ID parameter can be used to cause SyzMPF/z to use an alternate parmlib startup member in place of the SYZSUB00 one. For instance, PARM='ID=BW' would cause SyzMPF/z to use the parmlib member SYZMPFBW.

The DSN parameter can be used to tell SyzMPF/z what the Dataset name of the SCRIPT dataset is to be. This is normally set within the PARMLIB member (default SYZSUB00), but can be overridden by the use of this parm. This parameter will override the parmlib DSN= setting, and care should be taken that its use does not conflict with the dataset that the site normally wishes to use.

EXAMPLE JCL:

```
000001      //SYZMPFZ PROC
000002      //SYZMPFZ EXEC PGM=SYZSUBSS,TIME=1439,REGION=0M,
000003      // PARM='DSN=SYS1.SYZMPFZ.SCRIPTS'
Or
000001      //SYZMPFZ PROC
000002      //SYZMPFZ EXEC PGM=SYZSUBSS,TIME=1439,REGION=0M,
000003      // PARM='ID=BW'    ← this will use the SYZMPFBW parmlib member
```

The started task should have RACF authority to read the data set identified in the parameter or in the SYZMPFxx parmlib member as well as read the parmlib itself.

Required steps to process console ALL message traffic.

To process console message traffic, you will use the SyzMPF/z MPF exit program (normally **SYZMPFZ**). This program was delivered with your base XMIT data set and should be copied to an authorized load library in LinkList.

In order to activate SyzMPF/z for a message there are two parts to complete, first, you need to tell the system what message(s) you wish SyzMPF/z to process. It's triggered by placing an entry in the active system parmlib MPF list (normally MPFLSTxx, where xx is the suffix controlled in the CONSOLnn member, the default is 00 (zero-zero)).

The second part is that a corresponding member of the Command Scripting data set is necessary. So what these two steps do is tell IBM's Message Processing Facility that you want SYZMPFZ to process a specific (or generic) message, and once MPF passes the message to SYZMPFZ, SYZMPFZ will branch to the memory copy of the "script" that you want to use to process that message, and do whatever it is you need done when that message occurs.

If the message to be acted upon is longer than the 8-character maximum (which can happen for messages now that they can contain up to 17 characters), the member name that is used is the first 8 bytes of the message ID.

For example let's assume the current SYS1.PARMLIB MPF list member is (MPFLST00) :
Contents of "SYS1.PARMLIB(MPFLST00)

Comments in the MPFLSTxx member begin with "/*" which stops all processing for that line only.

```
.....  
/*  
DFHSI1517,SUP(NO),USEREXIT(SYZMPFZ)  
/*  
/* The exit uses the message id as a member in the Parmlib  
/* concatenation. Using the example for message DFHSI1517,  
/* there will be an DFHSI151 (note only 8 characters are used) member in the  
/* Script Parmlib "SYS1.MPF.SCRIPTS(DFHSI151)" which will contain  
/* The script logic and possible commands that are to be issued  
/* in response to DFHSI1517.
```

Contents of "SYS1.MPF.SCRIPTS(DFHSI151)" (lines starting with "*" are comments)

```
*Process the DFHSI151x commands.  
* we want DFHSI1517 control is given to CICS
```

- * First check to see if it's the message we are looking for, if not, fall through to end:
- * the reason we do this is because the MPFLSTxx member might have come here
- * as a generic "DFH*" processing, (this example didn't but we wanted to show it)

IFMSGID DFHSI1517 ← begins an IF nest

- * if CICS123 start DB2 Connect A

IFMSGTASK CICS123 ← another IF nest, was this message issued by CICS123?

S DB2CONA ← YES, there is no ELSE, so we only do "something" if true

ENDIF ← end of this "inner" IF nest

- * if CICS456 start DB2 Connect B

- * note that the IF commands can be followed by spaces (or not)

IF MSGTASK CICS456 ← was this message issued by CICS456?

S DB2CONB ← YES, there is no ELSE, so we only do "something" if true

ENDIF ← end of this second "inner" IF nest

ENDIF ← end of the IF nest that started "IF MSGID" above

Some basic things to know:

There are some basic setup functions that you must first understand and implement in order to use SyzMPF/z (at all) which go beyond just installation of the software into a "LINKLIST" library. There are two important steps to using the product that all sites must perform.

We will be tossing some terms around within this book and we will try to define them on first use, but sometimes it will be convenient to simply re-define them over and over again so that the user doesn't need to find a glossary every time we use the terms. We hope this doesn't cause any problems, and hopefully we define them exactly the same way each time, but sometimes it can be confusing when this manual refers to something that we feel is a "common" name which might or might not be all that "common" for the reader. One example of this is the term "MESSAGES" which actually refers to both CONSOLE messages (those that actually "roll" across the operations console) and SYSLOG messages (those that are written to the JES2 syslog which may or may not (based on routing codes) also scroll on an operations console). There are "other" kinds of messages as well which are also able to be "seen" by SyzMPF/z, (JOBLOG messages) which are actually still CONSOLE or SYSLOG messages which "can" be limited to only displaying on a task's JOBLOG and not the CONSOLE or SYSLOG. SyzMPF/z can turn any CONSOLE or SYSLOG message into a JOBLOG message or turn CONSOLE messages into SYSLOG ONLY messages, or in fact delete the message as if it never existed, (but still act on it), but for the purposes of this definition "MESSAGES" will refer to the generic form of CONSOLE, or SYSLOG or JOBLOG messages equally.

The first step is to tell z/OS console message processing which messages you want SyzMPF/z to handle. Some "other" console management products, simply insert their code into z/OS such that EVERY console or SYSLOG message is processed by their code. The simple fact is that the vast majority of console and syslog traffic is completely worthless when it comes to system automation. Consider the fact that a "standard" (according to IBM) site generates well over 150,000 console or syslog messages, but most sites (all that we know of) normally only "care" about a VERY small fraction of those messages. Some automation products will install themselves so that they will be sent EACH and EVERY ONE of those messages, even though the vast majority of them are wasted effort. The main difference that makes SyzMPF/z so much more resource efficient than competitor products, is that instead of forcing every single console (or syslog) message to be processed by SyzMPF/z, we only process that small subset that the site wants "automated" the rest are never passed to SyzMPF/z to even look at. This saves z/OS overhead because the path-length of code to pass the unnecessary messages is quite long (for each message) and also because SyzMPF/z doesn't need to do anything for messages that it doesn't "see", so processing which would need to "decide" whether to do "anything" for that message is completely bypassed. No getmains, no freemains, no memory transfers, not a single instruction is necessary.

Compared to the “other” automation products, the overhead of SyzMPF/z is virtually unable to even be measured. You don’t have to give away 10 to 15% of your resources for automation, you can use SyzMPF/z and have the overhead virtually non-existent.

The way you go about having z/OS tell SyzMPF/z to “automate” a message is by adding an entry in the (SYS1.) PARMLIB MPFLSTxx member. The MPFLST is the PARMLIB member that controls how you (the site) want all messages to be “handled” by z/OS. In the case of SyzMPF/z you would merely tell z/OS which of the messages that you want z/OS to “pass” on to be handled by automation.

MPFLSTxx contains information that the message processing facility (MPF) uses to control all aspects of messages, such as:

Message Presentation

On certain devices, messages can appear with highlighting, in color, or with added intensity.

Message Management

Message management refers to message suppression, message retention, and message processing.

Message suppression

A suppressed message does not appear at a console but is written to the hard-copy log.

Message retention

The action message retention facility (AMRF) keeps action messages in a buffer area, allowing the operator to request that any action message not acted on be recalled to the screen. In MPFLSTxx, you can identify certain action messages that AMRF is not to retain.

Message processing

Use MPFLSTxx to identify messages to be processed either using a message automation subsystem (for example, NetView or using installation-written exits.

Command Processing

You can specify installation exits (in this case SyzMPF/z) that will get control each time a command is issued.

Helpful Syntax rules for MPFLSTxx

These rules apply to the creation of MPFLSTxx in (SYS1.) PARMLIB:

- Each record is 80 characters long. The system ignores columns 72 through 80 and leading blanks.
- Each message processing record (line of the member) can contain only one message identifier. The message processing record for msgid cannot have embedded blanks. You can have messages prefixes (i.e. some part of the message ID that ends with an asterisk), but even though actual MESSAGES can be almost any length, you can only use the **first 10** of them in MPFLSTxx, sad, but it's an IBM restriction. That can sometimes create some problems, but all you need to remember is that while MPFLSTxx can only differentiate between the first 10 characters of the message, SyzMPF/z can work with the entire message (up to 50 "words") but SyzMPF/z treats the first 25 (48 bytes if really necessary) bytes of the message (up to the first blank space) of that message ID as the "actual" message ID.

For instance, message **BPXRMMDFXS**D221 and **BPRRMMDFXS**R11A to MPFLSTxx are the same message because all it "sees" is "**BPXRMMDFXS**", so that's what you had to code as the MPFLSTxx messageid to get the message processed by automation in the first place. But when you pass this entry on to SyzMPF/z, you can process it in one of several ways to "weed" out the messages you don't care about, or just process the ones you do care about. (more on this later, but basically you can use the "IF MSGID =" command or the "&MSGID =" variable to decide what to (or not to) process). So just because IBM decided to make messages longer without updating their MPF facility to handle them doesn't keep you from processing them.

Actually this is a good thing as well, because it allows you to process ANYTHING that crosses the console or SYSLOG. This little "feature" allows you to make on the fly operator commands. For instance, you can allow the operator to enter "@END SHIFT" when their shift is done. The MPFLSTxx member can have an entry for @END (or even just @) and have SyzMPF/z process that "message" and execute a script, which can parse the "message" and see that it's a request to do "end of shift" stuff for this particular operator. Since SyzMPF/z can "see" the console that entered the command, we can tell who this operator is, and within the script, we can do whatever is necessary for this particular operator's end of shift, like changing Initiator classes, stopping or starting scheduled tasks, send email to literally "anyone" that the operator is deserting us, close some files or logs or open others and log things to them, the list is virtually endless, and is all

started by what (to the operator) appears to be a valid operator command, but which is actually one that the site "made up". We actually have a later example of this in this manual, where we set up the command "@" which has many (many) options, one of which is @IPL and when it's entered, it checks to see which LPAR it was entered on and will kick off a (in this case) SyzCMD/z script from SyzMPF/z that will first ask the operator if they "really" mean to IPL that particular LPAR, and if they do, it will gracefully shut everything down and quiesce the LPAR. There are (many) other options as well. @STATS will display all of the MPF messages that have been processed (since the last IPL) and how many of each one, (and by what script). @PAGE will send a page (using SyzMAIL/z) to whoever it is they wanted to page, with up to 110 characters of text. By "page" we are referring in this case to either (or both) an EMAIL and/or an SMS text message (Cell phone message), for specific personnel by name. there are many other options as well, there is no limit except for your imagination.

- The message processing record for .MSGCOLR and .MSGIDS must have a single blank between the statement and the operand. Embedded blanks are not allowed anywhere else on the message processing record.
- Begin comments with /* in any column. It is best to end the comment with */. If, however, you do not include the end delimiter, the system recognizes the next statement because it's on the next line, everything else on this line (after the "/*" is considered a comment.

Controlling message management

MPFLSTxx allows you to specify how you want to process WTO/WTOR messages.

Message management refers to message suppression, message retention, message automation eligibility, and the use of installation-supplied WTO/WTOR exit routines.

Message suppression means that the message is logged in the hard-copy log, but it does not appear at an MVS operator's console.

Message retention means that action messages are saved by the action message retention facility (AMRF) on the action message retention queue. Retention allows the operator to view the message later. If you choose not to retain a message, the system will not add it to the action message retention queue.

Message automation eligibility means that you are using an automation subsystem to process particular WTO/WTOR messages in a pre-determined way. The automation subsystem (such as SyzMPF/z) allows the installation to program the processing for a particular message. For example, your installation may be using an automation subsystem to:

- Modify the text of a message.
- Re-route a message to a different operator's console or to a different system for processing.
- Record some information contained in the message text for future use.
- Respond to the message.

An automation subsystem can look at the message text and the message attributes and perform the programmed action. You can use MPFLSTxx to identify whether you want a message to be eligible for automation processing. The automation subsystem must then select and process the messages. You can also use MPFLSTxx to indicate information to pass to an automation subsystem. Messages that are eligible for automation processing can also be suppressed or retained. Message suppression and retention are functions separate from automation.

Your installation might have installation-written exit routines to handle WTO/WTOR messages and such is the case with SyzMPF/z. Through MPFLSTxx, you can specify which messages are to go to which exit routine. The exit routine can examine the message text and the message attributes and decide whether to suppress, retain, or take other actions on the message. The exit routine can also examine and modify the token.

Special MPFLSTxx AUTO actions which can be taken with SyzMPF/z

AUTO() Token

The AUTO(token) parameter of MPFLSTxx allows you to pass control information on to SyzMPF/z and have it execute a different MPF script member than the "default" one that matches the MessageID of the message that caused the MPFLSTxx line to take effect. For instance, if you wanted to precess the SYZ1234I message with a SyzMPF/z script of SYZ1234I, you wouldn't need to code anything at all in the AUTO() parameter, but if you wanted instead to execute a script called "REPLYYES", you would code it as;

SYZ1234I,SUP(NO),USEREXIT(SYZMPFZ),AUTO(**REPLYYES**)

- Putting any un-restricted name in this field will cause SyzMPF/z to use the MPF Script member REPLYYES when message SYZ1234I is issued instead of the SYZ1234I member that SyzMPF/z would normally invoke.

- You can also tell SyzMPF/z to perform several message modifications as well as tell SyzMPF/z operate in "QUIET" mode while processing messages. "QUIET" mode, means that SyzMPF/z will not put out any "extra" messages to the HARDCOPY or JOBLOGs informing that it has processed a message automatically. Some sites don't want this extra message traffic, and SyzMPF/z provides the ability to NOT generate it.

All of the "**special**" processing commands are passed to SyzMPF/z via the AUTO(xxxxxxx) parameter. Those "special" commands are:

AUTO(QUIET)	-	Turn off automatic message notification
AUTO(SUPPRESS)	-	Suppress the MessageID from both SYSLOG and JOBLOG
AUTO(NOJOBLOG)	-	Suppress the MessageID from the JOBLOG only
AUTO(NOSYSLOG)	-	Suppress the MessageID from the Hardcopy (SYSLOG) only.
AUTO(HIGHLIGHT)	-	Highlight the MessageID on the Operator Console
AUTO(LOWLIGHT)	-	Lowlight a highlighted message on the operator console
AUTO(REVERSEV)	-	change message to reverse video
AUTO(UNDERLIN)	-	Underline the message on the console
AUTO(color)	-	Change color, can be RED, BLUE, PINK, YELLOW, GREEN, WHITE, TURQ
AUTO(anythingelse)-		Tells SyzMPFz to use an alternate MPF Script for this Message.

QuickReply with MPFLSTxx AUTO(#xxxx...) actions which can be taken with SyzMPF/z

AUTO(#xxxxxxxxxxxxxxxxx)

The AUTO(#token) parameter of MPFLSTxx allows you to pass WTOR reply information on to SyzMPF/z and have SyzMPF/z reply to the message with the exact text as it exists following the "#" character. For instance;

SYZ1234I,SUP(NO),USEREXIT(SYZMPFZ),AUTO(**#YES**)

- NOTE: Upper/lower case is maintained on the replies, but typically MVS console commands and replies are changed to upper case for most replies.

All of the “**special**” processing commands are passed to SyzMPF/z via the AUTO(#xxxxxxx) parameter. Those “special” commands are:

AUTO(#STOP)	-	Will issue “STOP” to the outstanding WTOR
AUTO(##STOP)	-	Will issue “#STOP” to the outstanding WTOR

*******NOTE******* Unlike SyzCMD/z, you **CAN NOT SEND EMAIL DIRECTLY** from SyzMPF/z without first installing and activating the SyzMAIL/z add-on product.

4. Global Settings (system parmlib)

A integral feature of SyzMPF/z is the inclusion of GLOBAL settings which are contained within a member of the system parmlib concatenation. The default parmlib member name is SYZSUB00 (zero-zero), and the suffix of this member can be controlled via the startup parm of SYZSUBSS of “ID=nn”. i.e.:

```
//SYZMPF EXEC PGM=SYZSUBSS,PARM='ID=A7'
```

will result in the parmlib member [SYZSUBA7](#) being used instead of SYZSUB00.

There are several new GLOBAL parameters that can be contained within the parmlib member that SyzMPF/z uses and they will be described in this section. These settings will be used for all invocations of the SyzMPF/z message processor. These settings are analyzed by the SYZSUBSS program and while every effort is made to edit the provided GLOBAL settings programmatically, care should be taken to view the output from SYZSUBSS when the settings are changed to make sure they are used in the way the site actually wanted them to be used. These settings are loaded into a memory area that is immediately accessible by the SyzMPF/z message processing facility without any physical i/o or memory access delays. The method used is extremely efficient, and the overhead of using the GLOBAL settings in this way are so low as to be considered trivial.

NOTE: In the following section, the commands are presented in UPPER and lower case. The reason for this is to show that you **MUST** code the UPPER case part of the command and the lower case part is only used for ease of visually reading a script. For instance, “ECHO=W” means to echo any warnings that occur within processing, and could be used as “ECHO=WARN” or “echo=warnings” or “EcHo=WaRnInG” they all mean exactly the same thing to SyzMPF/z, some are just easier for “real people” to read.

NOTE: When script global settings and command parameters are displayed, the “option(s)” that are **highlighted and underlined** are the default for that parameter. If no option is presented as such, then there is no “default” setting for that entry.

NOTE: Any lines that begin with a “*” (Asterisk) are considered comments, you can (**and should**) use as many of them as you think are necessary, they are not processed by SyzMPF/z and cause no ill effects or extra overhead in processing.

Sample SYZSUB00 "SYSTEM PARMLIB" DEFAULTS member

```

*
* This is the SAMPLE V5.6 SysSUBSS/SyzMPFz Startup parm member
*
*
* Debug=yes|NO          Debug SyzSUBss program execution
*                        Default is NO
*
* STATs=All|No|Defaults Collect Message processing Statistics
*                        Default is "Defaults"
*
* Mode=QUIet|NOQuiet    Show SYZMPF/z MSGid processing MSG
*                        default is NOQUIET)
*
* Mode=DEBug|NODEbug    Debug SyzSUB and SyzMPF execution
*                        (default is NODEBUG)
*
* Mode=TRAcE|NOTrace    Trace SyzMPFz program calls
*                        (default is NOTRACE)
*
* Mode=SIMulate|NOSIMulate simulate (do not issue) SyzMPFz commands
*                        (default is NOSIMULATE)
*
* Mode=VERBose|NOVerbose Longer informational messages
*                        (default is NOVERBOSE)
*
* ECHO=OFF or NONE      Echo NOTHING
*   =All                Echo Everything
*   =Commands|NOCommands Echo only SyzMPFz commands
*   =Ifs|NOIfs          Echo only IF based commands
*   =Sets|NOSets        Echo only SET/GET commands
*   =Bypassed|NOBypassed Echo Bypassed commands
*   =Global|NOGlobal    Echo Global information
*   =Warnings|NOWarnings Echo Warnings
*   =Parse|NOParse      Echo Parse data (by words)
*   =Errors|NOErrors    Echo Errors
*
* MAXMSG=9999           Maximum Email MSG length (if Syzmail/z)
*                        (default is 50)
*

```

```

* MPFDSN=dataset.name      SyzMPF/z Script DSN to use
*
* ECSAMAX=nnnK|M           Set the maximum ECSA size to use for holding
*                           the scripts from MPFDSN. (default=500K)
* SYZMAIL=Yes|No           SyzMAIL/z installed and active
*                           (default is NO)
*
* VERBOSE=Yes|No           Longer Informational messages
*                           (default is NO)
*
* SIMULATE=Yes|No          simulate (do not issue) SyzMPFz commands
*                           (default is NO)
*
* TRACE=Yes|No             Trace SyzMPFz program calls
*                           (default is NO)
*
* FIRST=member|@FIRST      Script member to execute before normal
*                           Scripts
*                           (default is @FIRST, if it exists)
*
* LAST=member|@LAST        Script member to execute AFTER normal
*                           Scripts
*                           (default is @LAST, if it exists)
*
* TO:Email Address         Default TO address
*
* CC:Email Address         Default CC address
*
* BCC:Email Address        Default BCC address
*
* FROM:Email Address        Default FROM address
*
* REPLYTO:Email Address     Default Return address
*
* TIMEZONE=+/-nnnn or USzone Time zone of this system (default is PST)
*
* SUBJECT:anything         Default Subject
*
* EFORMAT=html|text|both    Type of maxCC email output to produce
*
* * * * *
DEBUG=Yes

```

TRACE=No
MODE=NOSimulate
ECHO=NONE
MPFDSN=Sys1.SyzMPFz.Scripts
SYZMAIL=Yes
VERBOSE=Yes
TO: Beta@SyzygyInc.com
CC: NobodyCC@SyzygyInc.com
BCC: NobodyBCC@SyzygyInc.com
FROM: SyzMPFz@SyzygyInc.com
REPLYTO: WebSiteSupport@SyzygyInc.com
Subject: Message from SyzMPF/z
TIMEZONE= -0700
STATS=All

MPFDSN

= **Script.data.set.name**

The MPFDSN GLOBAL setting parameter is used to tell the SyzMPF/z Subsystem (SYZSUBSS) which script dataset to use for processing all messages. Previously, this parameter was required to be specified as the execution parm to the SYZSUBSS program, and while it still can be used in that way, it is strongly suggested that the site change to using this parmlib setting instead. Both the normal load processing, and the actual Syzygy SUBSYSTEM, (which is loaded at IPL time via an entry in parmlib member IEFSSNnn), will use this setting if provided. If it is not provided, then SYZSUBSS will attempt to look for the execution parameter, and if it is not found, an error will be generated.

There are no case restrictions, the dataset name can be any combination of upper and lower case as the entire DSN is rendered upper case before use. Linux files can be used for scripts in some of the other Syzygy Automation products, but not so with SyzMPF/z. This dataset **MUST** be a "standard" PDS (not PDS/e) and must have an LRECL of 80, the BLKSIZE is not relevant (to SyzMPF/z), so any efficient BLKSIZE is acceptable.

Over time, this dataset can (and probably will) end up with hundreds if not thousands of script members, so unless you are severely constrained for space, we suggest that you define the dataset as 50 to 100 (or more) cylinders with at least 250 directory blocks. That should allow well over 1,000 members with a lot of space left over. While there are PTF's available that will allow SyzMPF/z to use a PDE/e dataset, it is not recommended at this time. IBM doesn't yet support using PDS/e before loading the PDS/e support at IPL time and a standard PDS is adequate and functions with less overhead.

Each member of the PDS was limited to 100 lines of actual Scripting code (comments don't count), but we changed that with Version 5.6 of SyzMPF/z so now there is no restriction on number of lines in a script. We determined a long time ago that "most" scripts are under 20 lines long, but over time we added some features which made it advantageous to make scripts a bit longer. Version 5.6 loads the length of the script and the pointer to the "next" script entry as the first element of each script's memory area, so instead of EVERY script being the same length, each one can be any length with instant availability of the end and beginning (and next in line) of each entry available at all times. This was always possible with the use of the SKIPTO command, and there was no logical limit to the length of a "logical" script, even though physical scripts (actual PDS members) were limited to 100 lines (but no longer are limited at all), they can still be logically linked (via SKIPTO), so that there is no real limit to script processing. SKIPTO is still extremely beneficial in cases where you have a common set of code that you want to use in multiple scripts and don't want to have to code it

multiple times. Because all scripts are loaded into a memory space, there is no extra processing overhead involved in linking multiple script together via SKIPTO. The changes to the script length in version 5.6 are solely to address the usability aspect of being able to use a single PDS member instead of multiple PDS members for any given script and to save memory space allocated to the vast majority of scripts which use far less than the previously allocated 8,000 bytes for each script loaded from the script PDS named in "MPFDSN=".

Parameters:

Any supported PDS data.set.name



*note: This dataset MUST exist, and MUST be cataloged

Example:

.....SYS1.PARMLIB(SYZSUB00) member contents...

DSN=SYS1.SyzMPFz.scripts

ECHO=NONE

MODE=NOSIMULATE

MODE=NODEBUG

MODE=NOQUIET

MODE=VERBOSE

STATISTICS=Default

SYZMAIL=Yes

MAXMSG=100

To: AllMail@SyzygyInc.com

From: SyzMPFz@SyzygyInc.com

CC: JOBLOG@SyzygyInc.com

FIRST=@FIRST

LAST=@LAST

←Script dataset

←don't echo any commands by default

←Default setting

←Default setting

←Default setting

←Default setting

←Default setting

←Allow email to be generated

←Default is 25

←Default to use if script forgets or is invalid

←Authorized to send mail from this LPAR

←send copy to JOBLOGs

←Script DSN member of cmds to do first

←Script DSN member of cmds to do last

.....

ECSAMAX

= **nnnK or M** (K=Kilobytes, M=Megabytes)

The "ECSAMAX" GLOBAL setting parameter is used to tell the SyzMPF/z Subsystem (SYZSUBSS) how large an ECSA area will be necessary to store the scripts that will be used by SyzMPF/z. Each script is made up of from 1 to 99,999 80 byte lines of script commands within each script. This parameter controls how much memory will be used for storing those scripts. The maximum size allowed is 9 Megabytes, which would hold over 112,000 lines of script commands. There is a PTF available that would increase that to 99Megabytes, but it is not recommended that it be used under normal circumstances. When the script space is created, SyzMPF/z keeps track of how much is allocated compared to how much is actually used and will generate a message after all of the scripts are loaded which will tell you how much space is (wasted) unused. You can reduce or increase the size of the ECSA space as may be required. This space is in Extended CSA storage, and will have no effect on the amount of memory used by or for normal batch jobs or started tasks in any way.

Parameters:

nnn	1 to 3 digit number (max is 999, minimum is 1, default is 512K)
Kilobytes	Allocate in Kilobytes (the default)
Megabytes	Allocate in Megabytes (9 Megabytes is the limit)

Example:

.....SYS1.PARMLIB(SYZSUB00) member contents...

DSN=SYS1.SyzMPFz.scripts	←Script dataset
ECHO=NONE	←don't echo any commands by default
MODE=NOSIMULATE	←Default setting
MODE=NODEBUG	←Default setting
MODE=NOQUIET	←Default setting
MODE=VERBOSE	←Default setting
STATISTICS=Default	←Default setting
SYZMAIL=Yes	←Allow email to be generated
MAXMSG=100	←Default is 25
To: AllMail@SyzygyInc.com	←Default to use if script forgets or is invalid
From: SyzMPFz@SyzygyInc.com	←Authorized to send mail from this LPAR
CC: JOBLOG@SyzygyInc.com	←send copy to JOBLOGs
ECSAMAX=320K	←Script ECSA space maximum (320Kilobytes)
LAST=@LAST	←Script DSN member of cmds to do last

.....

ECHO

The ECHO GLOBAL setting parameter is used to tell the SyzMPF/z Subsystem (SYZSUBSS) what degree of command ECHOing is to take place (by default). This can be overridden by the regular script ECHO command, and is identical in format to that "run-time" command, except this GLOBAL parameter sets the default ECHO setting that SyzMPF/z will use for all script processing. It is not recommended that you set up excessive echoing (except maybe during initial testing of the product itself), and instead, when you are testing a "new" script or changes to one, that you use the "ECHO=" command within that script and not change the default for ALL SCRIPTS with the GLOBAL parameter. The operators will not think it's funny when they get a few thousand extra "things" floating by on the console.

The settings are cumulative, so you can add or subtract them in any order or in separate command requests (one per line). You may have as many of ECHO= lines in the startup member as you wish, but only ONE SETTING PER LINE is used from each one. If there are any "conflicting" settings, (i.e. "ECHO=ALL" followed by "ECHO=NONE"), the last one processed takes precedence.

Adding{

A | All
I | Ifs
W|Warnings
E|Errors
G|Global
B|Bypassed
P|Parse
C|Commands }

subtracting{

NONE or OFF
NOI |NOIfs
NOG|NOGlobal
NOW|NOWarnings
NOE|NOErrors
NOB |NOBypassed

NOP |NOParse
NOC|NOCommands }

The ECHO global control command is used to control whether or not and to what extent that the script commands, logic and issued commands are echoed to the console and syslog.

Parameters:

A | All
I | Ifs
W|Warnings
G|Global
B |Bypassed
P |Parsed
C|Commands
N | None or O|Off
NOI |NOIfs
NOG|NOGlobal
NOW|NOWarnings
NOB |NOBypassed
NOP |NOParse
NOC|NOCommands

A or All - Causes SyzMPF/z to ECHO all commands and script control logic to console.
****This was the default before Version 2.0, but is no longer suggested as a default

NONE or OFF - Is the opposite of ALL, it will eliminate all ECHOing of the commands and any errors (or warnings) that may occur.

G or Global - Causes SyzMPF/z to ECHO the global scripting commands to the console (currently those are limited to ECHO and EOSM). This is the default setting along with C.

NOGlobal - is the opposite of Global.

I or Ifs - Causes SyzMPF/z to ECHO IF-Related logic to the console and syslog, (IF STARTED, IF SYSID, ELSE, ENDIF etc.) This is something that as a “default” is not a “good thing” because you will end up echoing a LOT of information to the console/syslog. It’s much better to use the ECHO command in your script when you think it might be necessary.

NOIfs - is the opposite of Ifs.

C or Commands- Causes SyzMPF/z to ECHO the issued system commands to the console and syslog, This is the default setting along with G. This is something that as a “default” is not a

“good thing” because you will end up echoing a LOT of information to the console/syslog. It’s much better to use the ECHO command in your script when you think it might be necessary.

NOCommands - is the opposite of Commands.

W or Warnings - Causes SyzMPF/z to ECHO Warnings that may occur during the execution of any script to the console and syslog. Warnings are for relatively minor things like forgetting to use an equal sign or not specifying a start position for something. In these cases, SyzMPF/z will “assume” some things to be valid, and on rare occasions that might not be what you want. For the most part, setting Warnings as a default is not very useful, it’s much better to add an ECHO command to the script, rather than force ALL scripts to produce warnings.

NOWarning - is the opposite of Warnings. *** Note: Currently not supported

B or Bypass - Causes SyzMPF/z to Bypassed-Related logic to the console and syslog, (IF clauses that fail, ELSE clauses that fail, etc.) This is not really something that you want to do as a matter of course. You are actually telling SyzMPF/z that all logic in all scripts that gets bypassed because it didn’t apply to your current situation. It can be helpful in some very long scripts or if you are trying to figure out why something was NOT executed, but in those cases, it’s much better to use the ECHO command within the individual script.

NOBypass - is the opposite of Bypass.

P or Parse - Causes SyzMPF/z to send the individual parsed out words (one per line and numbered as to the corresponding WORD number of the message) in response to the PARSE2WORDS command on the console. This is more helpful when invoked as an ECHO command within individual scripts, because you will otherwise be displaying all words of automation processed messages on the console (one line per word).

NOParse - is the opposite of Parse.

Example:

.....SYS1.PARMLIB(SYZSUB00) member contents...

DSN=SYS1.SyzMPFz.scripts	←Script dataset
ECHO=NONE	←don't echo any commands by default
MODE=NOSIMULATE	←Default setting
MODE=NODEBUG	←Default setting
MODE=NOQUIET	←Default setting
MODE=VERBOSE	←Default setting

STATISTICS=Default	←Default setting
SYZMAIL=Yes	←Allow email to be generated
MAXMSG=100	←Default is 25
To: AllMail@SyzygyInc.com	←Default to use if script forgets or is invalid
From: SyzMPFz@SyzygyInc.com	←Authorized to send mail from this LPAR
CC: JOBLOG@SyzygyInc.com	←send copy to JOBLOGs
FIRST=@FIRST	←Script DSN member of cmds to do first
LAST=@LAST	←Script DSN member of cmds to do last

.....

DEBUG

This would normally only be used under direction of Syzygy Product Support. The "DEBUG" parameter is used to tell SyzMPF/z whether or not to display DEBUGging information on the console and in the LOG. This command is used to turn off some debugging information such as the creation and deletion of email packages and other messages which may not be desired by the site. This is not the same as "QUIET" although it is similar. **This GLOBAL parameter can also be used as a script command.**

When debugging is turned on, SyzMPF/z will trace every enter and exit to every module and sub-routine executed within a script. This would be a very bad thing to do as a "default" for all scripts, it's MUCH better to use this as a command within an individual script, the number of lines created can be quite extensive, even for small scripts. It's best to limit its use to sections of scripts when possible. Turning it on and off only for the areas you really want to debug.

Parameters:

Yes | **No** (no is the default)

Example:

Example:

.....SYS1.PARMLIB(SYZSUB00) member contents...

DSN=SYS1.SyzMPFz.scripts	←Script dataset
ECHO=NONE	←don't echo any commands by default
MODE=NOSIMULATE	←Default setting
MODE=NODEBUG	←Default setting
MODE=NOQUIET	←Default setting
MODE=VERBOSE	←Default setting
STATISTICS=Default	←Default setting
DEBUG=YES	← Show debugging messages
SYZMAIL=Yes	←Allow email to be generated
MAXMSG=100	←Default is 25
To: AllMail@SyzygyInc.com	←Default to use if script forgets or is invalid
From: SyzMPFz@SyzygyInc.com	←Authorized to send mail from this LPAR
CC: JOBLOG@SyzygyInc.com	←send copy to JOBLOGs
FIRST=@FIRST	←Script DSN member of cmds to do first
LAST=@LAST	←Script DSN member of cmds to do last
.....	

(email related) EFORMAT | EMFormat

= HTML | Text | NOHTML | NOText | Both

The EFORMATs start-up parameter tells SyzMPF/z (and SyzMAIL/z) what format to use for creating the email package to be delivered, HTML, TEXT or BOTH (HTML & TEXT).

This can be changed within the actual script, if it is deemed necessary that sometimes one format or the other pertains “on the fly”. Some email clients do not accept HTML format email, and some email clients can turn off HTML processing. Sending both formats guarantees that the client will get the information, (HTML will then only be displayed if supported by that client). Using “BOTH” will double the actual size of the email package and will take up more space, both memory and IP packets which need to be sent.

You can place the EFORMAT command anywhere within a normal command script, but logically it makes most sense to have it in the system parmlib GLOBAL member, or as the very first parameter of the script.

Parameters:

HTML	Send ONLY the HTML content (NOHTML turns this option off)
BOTH	The DEFAULT, sends BOTH the HTML & TET formats in the same email
TEXT	Send ONLY the TEXT content (NOText turns this option off)

Example:

.....SYSL.PARMLIB(SYZSUB00) member contents...

DSN=SYS1.SyzMPFz.scripts	←Script dataset
ECHO=NONE	←don't echo any commands by default
MODE=NOSIMULATE	←Default setting
MODE=NODEBUG	←Default setting
MODE=NOQUIET	←Default setting
MODE=VERBOSE	←Default setting
EFORMAT=BOTH	←Default setting
SYZMAIL=Yes	←Allow email to be generated
MAXMSG=100	←Default is 25
To: AllMail@SzygyInc.com	←Default to use if script forgets or is invalid
From: SyzMPFz@SzygyInc.com	←Authorized to send mail from this LPAR
Cc: JOBLOG@SzygyInc.com	←send copy to JOBLOGs
FIRST=@FIRST	←Script DSN member of cmds to do first
LAST=@LAST	←Script DSN member of cmds to do last
.....	

(email related) SYZMAIL

The SYZMAIL GLOBAL setting parameter is used to tell the SyzMPF/z Subsystem (SYZSUBSS) whether or not the sending of EMAIL (and SMS texts) from SyzMPF/z will be supported. This can be overridden by the script SYZMAIL command, and is identical to that command, except this GLOBAL parameter sets the default EMAIL setting that SyzMPF/z will use for all script processing

Parameters:

- Yes SyzMail/z will be used to send email from SyzMPF/z
No SyzMail/z will not be used, Email will not be able to be sent from SyzMPF/z
 "NO" is the default

Example:

.....SYS1.PARMLIB(SYZSUB00) member contents...

DSN=SYS1.SyzMPFz.scripts	←Script dataset
ECHO=NONE	←don't echo any commands by default
MODE=NOSIMULATE	←Default setting
MODE=NODEBUG	←Default setting
MODE=NOQUIET	←Default setting
MODE=VERBOSE	←Default setting
STATISTICS=Default	←Default setting
SYZMAIL=Yes	←Allow email to be generated
MAXMSG=100	←Default is 25
To: AllMail@SyzygyInc.com	←Default to use if script forgets or is invalid
From: SyzMPFz@SyzygyInc.com	←Authorized to send mail from this LPAR
CC: JOBLOG@SyzygyInc.com	←send copy to JOBLOGs
FIRST=@FIRST	←Script DSN member of cmds to do first
LAST=@LAST	←Script DSN member of cmds to do last

.....

(email related) MAXMSG

= nnnn (max value is 9999) (Default is 25)

The MAXMSG GLOBAL setting parameter is used to tell the SyzMPF/z Subsystem (SYZSUBSS) the maximum size (number of lines) that SyzMPF/z can use to generate an email or SMS text message. The default is 25 lines, which is normal for most dynamically generated email. If the special MaxCC and StepCC email command is used, you should know that each step of the generated email will use up one line, therefore you cannot send a 200 step job's condition codes if you leave the MAXMSG setting at the default (25). When the maximum number of lines in the email is reached, the processing continues, but the lines generated will not be sent. A message at the bottom of the email will alert the recipient that the mail message was truncated. This can be overridden by the script MAXMSG command on a per script basis and this default setting is identical to that command, except this GLOBAL parameter sets the default EMAIL setting that SyzMPF/z will use for all script processing

Parameters:

nnnn The number of lines to set aside for EMAIL or SMS text message content.
25 The default

Example:

.....SYS1.PARMLIB(SYZSUB00) member contents...

DSN=SYS1.SyzMPFz.scripts	←Script dataset
ECHO=NONE	←don't echo any commands by default
MODE=NOSIMULATE	←Default setting
MODE=NODEBUG	←Default setting
MODE=NOQUIET	←Default setting
MODE=VERBOSE	←Default setting
STATISTICS=Default	←Default setting
SYZMAIL=Yes	←Allow email to be generated
MAXMSG=100	←Default is 25
To: AllMail@SzygyInc.com	←Default to use if script forgets or is invalid
From: SyzMPFz@SzygyInc.com	←Authorized to send mail from this LPAR
CC: JOBLOG@SzygyInc.com	←send copy to JOBLOGs
FIRST=@FIRST	←Script DSN member of cmds to do first
LAST=@LAST	←Script DSN member of cmds to do last
.....	

(email related) MAXSTEPS

= nnnn (max value is 9999) (Default is 25)

The MAXSTEPS GLOBAL setting parameter is used to tell the SyzMPF/z Subsystem (SYZSUBSS) how many steps there will be (max) when computing the maximum number of lines (composed of MAXMSG, MAXSTEPSx2 (HTML and TEXT) and Default messages (25)) that SyzMPF/z can use to generate an email or SMS text message for this particular message packet. The default is 25 steps, which is about the average number of steps in a job/task. If the special MaxCC and StepCC email command is used, you should know that each step of the generated email will use up one line, therefore you cannot send a 200 step job's condition codes if you leave the MAXSTEPS setting at the default (25). When the maximum number of lines in the email is reached, the processing continues, but the lines generated will not be sent. A message at the bottom of the email will alter the recipient that the mail message was truncated. This can be overridden by the script MAXSTEPS command on a per script basis and this default setting is identical to that command, except this GLOBAL parameter sets the default EMAIL setting that SyzMPF/z will use for all script processing

Parameters:

nnnn The number of STEP lines to set aside for package message content.
25 The default

Example:

.....SYSL.PARMLIB(SYZSUB00) member contents...

DSN=SYS1.SyzMPFz.scripts	←Script dataset
ECHO=NONE	←don't echo any commands by default
MODE=NOSIMULATE	←Default setting
MODE=NODEBUG	←Default setting
MODE=NOQUIET	←Default setting
MODE=VERBOSE	←Default setting
STATISTICS=Default	←Default setting
SYZMAIL=Yes	←Allow email to be generated
MAXSTEPS=100	←Default is 25
To: AllMail@SyzygyInc.com	←Default to use if script forgets or is invalid
From: SyzMPFz@SyzygyInc.com	←Authorized to send mail from this LPAR
CC: JOBLOG@SyzygyInc.com	←send copy to JOBLOGs
FIRST=@FIRST	←Script DSN member of cmds to do first
LAST=@LAST	←Script DSN member of cmds to do last
.....	

SCRIPTMAX

= nnnn (max value is 9999) (Default is 50)

The SCRIPTMAX GLOBAL setting parameter is used to tell the SyzMPF/z Subsystem (SYZSUBSS) the maximum number of Scripts that may be concatenated for a single message being processed. SyzMPF/z allows SKIPTO support to handle the conditions where the 100 line limit on script sizes is an issue, but with the advent of the GOTO and ++INCLUDE commands and especially GOTO, it is possible to inadvertently create a LOOP. This parameter will limit the total number of scripts which can be concatenated.

Parameters:

nnnn The maximum number of script member concatenations allowed .
50 The default

Example:

.....SYS1.PARMLIB(SYZSUB00) member contents...

DSN=SYS1.SyzMPFz.scripts	←Script dataset
ECHO=NONE	←don't echo any commands by default
MODE=NOSIMULATE	←Default setting
MODE=NODEBUG	←Default setting
MODE=NOQUIET	←Default setting
MODE=VERBOSE	←Default setting
STATISTICS=Default	←Default setting
SYZMAIL=Yes	←Allow email to be generated
MAXMSG=100	←Default is 25
SCRIPTMAX=50	←Default is 50
To: AllMail@SyzygyInc.com	←Default to use if script forgets or is invalid
From: SyzMPFz@SyzygyInc.com	←Authorized to send mail from this LPAR
CC: JOBLOG@SyzygyInc.com	←send copy to JOBLOGs
FIRST=@FIRST	←Script DSN member of cmds to do first
LAST=@LAST	←Script DSN member of cmds to do last
.....	

FIRST

= **Script name** (“@FIRST” is the default)

The “FIRST” GLOBAL setting parameter is used to tell the SyzMPF/z Subsystem (SYZSUBSS) what command script is to be executed before every other script which is processed by SyzMPF/z. Previous versions of SyzMPF/z used the \$\$\$\$DFLT script for this type of processing, but it was found that the overhead of that method was quite high. The new “FIRST” GLOBAL setting parameter will only be used when an actual SCRIPT is processed, (i.e. not for simple processing such as NOSYSLOG, COLORING or HIGHLIGHT, etc.) The default member name for this is @FIRST, and if no FIRST= parameter is coded, but a member named @FIRST is located in the script PDS, then SyzMPF/z will automatically use that member and execute it before all other script processing for every script that is executed. This parameter is used in cases where a site might want to have a default setting that is not yet supported by SyzMPF/z, or if there is a message that the site might wish to have generated for every invocation of a script. If a member is designated, but does not exist within the script library, then this feature is ignored.

Parameters:

membername	name within the script library to use for processing.
@FIRST	(the default script name)

Example:

.....SYS1.PARMLIB(SYZSUB00) member contents...

DSN=SYS1.SyzMPFz.scripts	←Script dataset
ECHO=NONE	←don't echo any commands by default
MODE=NOSIMULATE	←Default setting
MODE=NODEBUG	←Default setting
MODE=NOQUIET	←Default setting
MODE=VERBOSE	←Default setting
STATISTICS=Default	←Default setting
SYZMAIL=Yes	←Allow email to be generated
MAXMSG=100	←Default is 25
To: AllMail@SzygyInc.com	←Default to use if script forgets or is invalid
From: SyzMPFz@SzygyInc.com	←Authorized to send mail from this LPAR
CC: JOBLOG@SzygyInc.com	←send copy to JOBLOGs
FIRST=@FIRST	←Script DSN member of cmds to do first
LAST=@LAST	←Script DSN member of cmds to do last
.....	

LAST

= **Script name** (“@LAST” is the default)

The “LAST” GLOBAL setting parameter is used to tell the SyzMPF/z Subsystem (SYZSUBSS) what command script is to be executed AFTER every other script which is processed by SyzMPF/z. Previous versions of SyzMPF/z used the \$\$\$\$\$\$T script for this type of processing, but it was found that the overhead of that method was quite high. The new “LAST” GLOBAL setting parameter will only be used when an actual SCRIPT is processed, (i.e. not for simple processing such as NOSYSLOG, COLORING or HIGHLIGHT, etc.) The default member name for this is @LAST, and if no LAST= parameter is coded, but a member named @LAST is located in the script PDS, then SyzMPF/z will automatically use that member and execute it AFTER all other script processing for every script that is executed. This parameter is used in cases where a site might want to have a default setting that is not yet supported by SyzMPF/z, or if there is a message that the site might wish to have generated for every invocation of a script. If a member is designated, but does not exist within the script library, then this feature is ignored.

Parameters:

membername	name within the script library to use for processing.
<u>@LAST</u>	(the default script name)

Example:

.....SYS1.PARMLIB(SYZSUB00) member contents...

DSN=SYS1.SyzMPFz.scripts	←Script dataset
ECHO=NONE	←don't echo any commands by default
MODE=NOSIMULATE	←Default setting
MODE=NODEBUG	←Default setting
MODE=NOQUIET	←Default setting
MODE=VERBOSE	←Default setting
STATISTICS=Default	←Default setting
SYZMAIL=Yes	←Allow email to be generated
MAXMSG=100	←Default is 25
To: AllMail@SzygyInc.com	←Default to use if script forgets or is invalid
From: SyzMPFz@SzygyInc.com	←Authorized to send mail from this LPAR
CC: JOBLOG@SzygyInc.com	←send copy to JOBLOGs
FIRST=@FIRST	←Script DSN member of cmds to do first
LAST=@LAST	←Script DSN member of cmds to do last

.....

(email related) To:**Email address or Nickname or &variable**

The "To:" GLOBAL setting parameter is used to tell the SyzMPF/z Subsystem (SYZSUBSS) what default TO: email address or nickname (nicknames are described in the SyzMAIL/z manual) to use for all outgoing eMail messages. This setting is not required and is only provided in the case where the script coder might forget to code a To: parameter within that script because that parameter is a required parameter for all outgoing eMail messages. This can be set to a variable such as "To: ¬ify", which will send the email or SMS text message to the nickname that matches the NOTIFY= parameter of the JOBcard or /* NOTIFY JECL parameter. Any valid variable may be used, but be sure to code a corresponding nickname in the SyzMAIL/z nicknames dataset (please see the SyzMail/z manual for more information on nicknames). This parameter was designed to be a safety catch, so if an email address is used, it should be one that can easily accept the email messages that have been coded in error, in that someone will actually see them and be able to correct the error in the SyzMPF/z script. (i.e. TO: SCRIPT_Errors@thissite.com .

Parameters:

Email Address or Nickname or &Variable

Example:

.....SYS1.PARMLIB(SYZSUB00) member contents...

DSN=SYS1.SyzMPFz.scripts	←Script dataset
ECHO=NONE	←don't echo any commands by default
MODE=NOSIMULATE	←Default setting
MODE=NODEBUG	←Default setting
MODE=NOQUIET	←Default setting
MODE=VERBOSE	←Default setting
STATISTICS=Default	←Default setting
SYZMAIL=Yes	←Allow email to be generated
MAXMSG=100	←Default is 25
To: AllMail@SzygyInc.com	←Default to use if script forgets or is invalid
From: SyzMPFz@SzygyInc.com	←Authorized to send mail from this LPAR
CC: JOBLOG@SzygyInc.com	←send copy to JOBLOGs
FIRST=@FIRST	←Script DSN member of cmds to do first
LAST=@LAST	←Script DSN member of cmds to do last
.....	

(email related) From:**Email address or Nickname or &variable**

The "From:" GLOBAL setting parameter is used to tell the SyzMPF/z Subsystem (SYZSUBSS) what default From: email address or nickname (nicknames are described in the SyzMAIL/z manual) to use for all outgoing eMail messages. This setting is not required and is only provided in the case where the script coder might forget to code a From: parameter within that script because that parameter is a required parameter for all outgoing eMail messages. This can be set to a variable such as "From: ¬ify", which will send the email or SMS text message from the nickname that matches the NOTIFY= parameter of the JOBcard or /* NOTIFY JECL parameter. Any valid variable may be used, but be sure to code a corresponding nickname in the SyzMAIL/z nicknames dataset (please see the SyzMail/z manual for more information on nicknames). This parameter was designed to be a safety catch, so if an email address is used, it should be one that can easily accept the email messages that have been coded in error, in that someone will actually see them and be able to correct the error in the SyzMPF/z script.

(i.e. From: SyzMPF/z@thissite.com .

Parameters:

Email Address or Nickname or &Variable

Example:

.....SYS1.PARMLIB(SYZSUB00) member contents...

DSN=SYS1.SyzMPFz.scripts	←Script dataset
ECHO=NONE	←don't echo any commands by default
MODE=NOSIMULATE	←Default setting
MODE=NODEBUG	←Default setting
MODE=NOQUIET	←Default setting
MODE=VERBOSE	←Default setting
STATISTICS=Default	←Default setting
SYZMAIL=Yes	←Allow email to be generated
MAXMSG=100	←Default is 25
To: AllMail@SyzygyInc.com	←Default to use if script forgets or is invalid
From: SyzMPFz@SyzygyInc.com	←Authorized to send mail from this LPAR
CC: JOBLOG@SyzygyInc.com	←send copy to JOBLOGs
FIRST=@FIRST	←Script DSN member of cmds to do first
LAST=@LAST	←Script DSN member of cmds to do last
.....	

(email related) Replyto:**Email address or Nickname or &variable**

The "ReplyTo:" GLOBAL setting parameter is used to tell the SyzMPF/z Subsystem (SYZSUBSS) what default ReplyTo: email address or nickname (nicknames are described in the SyzMAIL/z manual) to use for all outgoing eMail messages. This setting is not required and is only provided in the case where the script coder might want the email to have a different Reply-To setting than the From: setting for all outgoing eMail messages. This can be set to a variable such as "Replyto: ¬ify", which will send the email or SMS text message and set the Replyto email parameter to the nickname that matches the NOTIFY= parameter of the JOBcard or /* NOTIFY JECL parameter. Any valid variable may be used, but be sure to code a corresponding nickname in the SyzMAIL/z nicknames dataset (please see the SyzMail/z manual for more information on nicknames). This parameter was designed to be a safety catch, so if an email address is used, it should be one that can easily accept the email messages that have been coded in error, in that someone will actually see them and be able to correct the error in the SyzMPF/z script.

(i.e. Replyto: Default@thissite.com .

Parameters:

Email Address or Nickname or &Variable

Example:

.....SYS1.PARMLIB(SYZSUB00) member contents...

DSN=SYS1.SyzMPFz.scripts	←Script dataset
ECHO=NONE	←don't echo any commands by default
MODE=NOSIMULATE	←Default setting
MODE=NODEBUG	←Default setting
MODE=NOQUIET	←Default setting
MODE=VERBOSE	←Default setting
STATISTICS=Default	←Default setting
SYZMAIL=Yes	←Allow email to be generated
MAXMSG=100	←Default is 25
To: AllMail@SyzygyInc.com	←Default to use if script forgets or is invalid
Replyto: Default@SyzygyInc.com	←Authorized to receive mail from outside LPAR
CC: JOBLOG@SyzygyInc.com	←send copy to JOBLOGs
FIRST=@FIRST	←Script DSN member of cmds to do first
LAST=@LAST	←Script DSN member of cmds to do last
.....	

(email related) Cc:**Email address or Nickname or &variable**

The "Cc:" GLOBAL setting parameter is used to tell the SyzMPF/z Subsystem (SYZSUBSS) what default Cc: email address or nickname (nicknames are described in the SyzMAIL/z manual) to use for all outgoing eMail messages. This setting is not required and is only provided in the case where the script coder might want to make sure that a copy of the email is always sent to some default inbox, such as a production JOB execution log mailbox. This can be set to a variable such as "Cc: ¬ify", which will send a copy of the email or SMS text message to the nickname that matches the NOTIFY= parameter of the JOBcard or /* NOTIFY JECL parameter. Any valid variable may be used, but be sure to code a corresponding nickname in the SyzMAIL/z nicknames dataset (please see the SyzMail/z manual for more information on nicknames). This parameter was designed to be a safety catch, so if an email address is used, it should be one that can easily accept the email messages that have been coded in error, in that someone will actually see them and be able to correct the error in the SyzMPF/z script.
(i.e. Cc: JobLog@thissite.com .

Parameters:

Email Address or Nickname or &Variable

Example:

.....SYS1.PARMLIB(SYZSUB00) member contents...

DSN=SYS1.SyzMPFz.scripts	←Script dataset
ECHO=NONE	←don't echo any commands by default
MODE=NOSIMULATE	←Default setting
MODE=NODEBUG	←Default setting
MODE=NOQUIET	←Default setting
MODE=VERBOSE	←Default setting
STATISTICS=Default	←Default setting
SYZMAIL=Yes	←Allow email to be generated
MAXMSG=100	←Default is 25
To: AllMail@SyzygyInc.com	←Default to use if script forgets or is invalid
From: SyzMPFz@SyzygyInc.com	←Authorized to send mail from this LPAR
Cc: JOBLOG@SyzygyInc.com	←send copy to JOBLOGs
FIRST=@FIRST	←Script DSN member of cmds to do first
LAST=@LAST	←Script DSN member of cmds to do last
.....	

(email related) Bcc:

Email address or Nickname or &variable

The "Bcc:" GLOBAL setting parameter is used to tell the SyzMPF/z Subsystem (SYZSUBSS) what default (Blind Copy) Bcc: email address or nickname (nicknames are described in the SyzMAIL/z manual) to use for all outgoing eMail messages. This setting is not required and is only provided in the case where the script coder might want to make sure that a copy of the email is always sent to some default inbox, such as a production JOB execution log mailbox. This can be set to a variable such as "Bcc: ¬ify", which will send a copy of the email or SMS text message to the nickname that matches the NOTIFY= parameter of the JOBcard or /* NOTIFY JECL parameter. Any valid variable may be used, but be sure to code a corresponding nickname in the SyzMAIL/z nicknames dataset (please see the SyzMail/z manual for more information on nicknames). This parameter was designed to be a safety catch, so if an email address is used, it should be one that can easily accept the email messages that have been coded in error, in that someone will actually see them and be able to correct the error in the SyzMPF/z script.

(i.e. Bcc: JobLog@thissite.com .

Parameters:

Email Address or Nickname or &Variable

Example:

.....SYS1.PARMLIB(SYZSUB00) member contents...

DSN=SYS1.SyzMPFz.scripts	←Script dataset
ECHO=NONE	←don't echo any commands by default
MODE=NOSIMULATE	←Default setting
MODE=NODEBUG	←Default setting
MODE=NOQUIET	←Default setting
MODE=VERBOSE	←Default setting
STATISTICS=Default	←Default setting
SYZMAIL=Yes	←Allow email to be generated
MAXMSG=100	←Default is 25
To: AllMail@SyzygyInc.com	←Default to use if script forgets or is invalid
From: SyzMPFz@SyzygyInc.com	←Authorized to send mail from this LPAR
Bcc: JOBLOG@SyzygyInc.com	←send Blind copy to JOBLOGs
FIRST=@FIRST	←Script DSN member of cmds to do first
LAST=@LAST	←Script DSN member of cmds to do last
.....	

(email related) Subject: | Subj:

Any text or &variable

The "Subject:" GLOBAL setting parameter is used to tell the SyzMPF/z Subsystem (SYZSUBSS) what default subject: to use for all outgoing eMail messages if none is provided by the script. The default if this parm is not included is "Subject: No Subject Provided".

Parameters:

Any text or &Variables (up to 72 bytes)

Example:

.....SYS1.PARMLIB(SYZSUB00) member contents...

DSN=SYS1.SyzMPFz.scripts	←Script dataset
ECHO=NONE	←don't echo any commands by default
MODE=NOSIMULATE	←Default setting
MODE=NODEBUG	←Default setting
MODE=NOQUIET	←Default setting
MODE=VERBOSE	←Default setting
STATISTICS=Default	←Default setting
SYZMAIL=Yes	←Allow email to be generated
MAXMSG=100	←Default is 25
Subject: &MSGID related information	← Default in case none was supplied
To: <u>AllMail@SyzygyInc.com</u>	←Default to use if script forgets or is invalid
From: <u>SyzMPFz@SyzygyInc.com</u>	←Authorized to send mail from this LPAR
Bcc: <u>JOBLOG@SyzygyInc.com</u>	←send Blind copy to JOBLOGs
FIRST=@FIRST	←Script DSN member of cmds to do first
LAST=@LAST	←Script DSN member of cmds to do last
.....	

(email related) TimeZone or TZ =

Time zone offset to use (default = PST)

The "TimeZone or TZ" GLOBAL setting parameter is used to tell SyzMPF/z what default Time Zone to use for all outgoing eMail messages. This setting is not required but is suggested because the default is PST. Any valid offset may be used, including the standard US time zone offsets, PST,PDT,EDT,EST,CST,CDT,etc.) The site can also specify the offsets to the west or (- negative) or east (+positive) of GMT. i.e. -0700 is the pacific time zone (PST) west 7 hrs.

Parameters:

Timezone or offset (+/-) Default is PST (TZ=-0700)

Example:

.....SYS1.PARMLIB(SYZSUB00) member contents...

DSN=SYS1.SyzMPFz.scripts	←Script dataset
ECHO=NONE	←don't echo any commands by default
MODE=NOSIMULATE	←Default setting
MODE=NODEBUG	←Default setting
MODE=NOQUIET	←Default setting
MODE=VERBOSE	←Default setting
STATISTICS=Default	←Default setting
SYZMAIL=Yes	←Allow email to be generated
TZ = -0700	← Time offset is west 0700
MAXMSG=100	←Default is 25
To: AllMail@SzygyInc.com	←Default to use if script forgets or is invalid
From: SyzMPFz@SzygyInc.com	←Authorized to send mail from this LPAR
Cc: JOBLOG@SzygyInc.com	←send copy to JOBLOGs
FIRST=@FIRST	←Script DSN member of cmds to do first
LAST=@LAST	←Script DSN member of cmds to do last
.....	

(email related) ATTACH

JCL | JESJCL | JESMSG LG | JESYSMSG | DSN

This global command controls what part(s) of the JES output of the task being processed should be sent as an attachment to the email that is being generated. Multiple ATTACH parameters can be used in the start up, but each one can only have one option defined. There is also a “normal” script command that allows for controlling the attachments, and it’s probably much better to use it there, but it’s possible that you might want all emails to also include some part of the JOB, so it’s also provided here. You can also attach any dataset via DSN=datasetname or DSN=PDSdatasetname(member) to send either a sequential dataset or a member of a PDS.

Parameters:

JESJCL - Attaches the JCL images as output from the converter

JESMSG LG - Attaches the JES message log (WTO’s issued by the task)

JESYSMSG - Attaches the system messages (JCL resolution messages and condition codes)

JESJCLIN - Attaches the JCL as submitted or started. (JCL before JES processing)

DSN - Attaches ANY dataset or PDS member (but not complete PDS) named.

Example:

.....SYS1.PARMLIB(SYZSUB00) member contents...

DSN=SYS1.SyzMPFz.scripts	←Script dataset
ECHO=NONE	←don’t echo any commands by default
MODE=NOSIMULATE	←Default setting
MODE=NODEBUG	←Default setting
MODE=NOQUIET	←Default setting
MODE=VERBOSE	←Default setting
ATTACH=JCL	←Send the JCL images along with any email
SYZMAIL=Yes	←Allow email to be generated
MAXMSG=100	←Default is 25
To: AllMail@SzygyInc.com	←Default to use if script forgets or is invalid
From: SyzMPFz@SzygyInc.com	←Authorized to send mail from this LPAR
Cc: JOBLOG@SzygyInc.com	←send copy to JOBLOGs

FIRST=@FIRST

←Script DSN member of cmds to do first

LAST=@LAST

←Script DSN member of cmds to do last

.....

Mode Commands:

SyzMPF/z supports as of version 2.0 "MODE" processing. This allows the site to change the general processing mode that the facility will use. There are several available processing modes, and it is expected that more will be added over time.

MODE=SIMulate | NOSIMulate

This mode command will place the processing mode of the script being processed IN or OUT of simulation mode.

For instance, the site may decide that an individual command within a script, a whole script or ALL SyzMPF/z processing should be “simulated” so that while the script is processed as normal, the commands that would be issued from the script are “:simulated” to the console. This means that commands and replies to outstanding WTOR’s are issued as a WTO instead of an actual system command.

You can place the MODE=SIMulate or MODE=NOSIMulate command anywhere within a normal command script. If you want MODE=SIMulate or MODE=NOSIMulate to be “globally” active, (i.e. for all command scripts that SyzMPF/z uses), then you should place the MODE= command within the SYZMPFxx system parmlib member.

Parameters:

SIMulate or NOSIMulate

Example:

```
.....
* Make sure it's before 8am or after 5pm  after 5pm only simulate the commands.
IF BEFORE 08:00
  $P PRT15
  $PI
  $TI1-5,ABCD
  $SI1-5
ELSE
  IF AFTER 17:00
    MODE=SIMULATE
    $PI
    $TI1-5,CDEF
    $SI1-5
    MODE=NOSIM
  ENDIF
.....
```

MODE=QUIET | NOQUIET

This mode command will place the processing mode of the script being processed IN or OUT of QUIET mode. QUIET mode forces SyzMPF/z to NOT display the “SyzMPF/z is processing SCRIPTID” message and turns off all ECHO processing, which is similar to ECHO=OFF processing.

For instance, the site may decide that an individual section within a script, a whole script or ALL SyzMPF/z processing should be performed in “QUIET” mode so that while the script is processed as normal, and commands are issued normally, that all ECHO processing and any WTO’s that would be generated by SyzMPF/z are skipped

You can place the MODE=QUIET or MODE=NOQUIET command anywhere within a normal command script. If you want MODE=QUIET or MODE=NOQUIET to be “globally” active, (i.e. for all command scripts that SyzMPF/z uses), then you should place the MODE= command within the SYZMPFxx system parmlib member.

Parameters:

QUIET or NOQUIET

NOTE: ECHO=ALL will perform the function of MODE=NOQUIET, but MODE=NOQUIET will NOT set ECHO=all or any ECHO= function at all.

Example:

```
.....
MODE=QUIET /* so that SyzMPF/z doesn't issue "processing" messages */
$PI12
$TI12,AB
$SI12
MODE=NOQUIET
.....
```

STATistics | Stats

= All | Defaults | No

The STATs startup paramter will place the processing mode of the script being processed IN or OUT of statistics collection mode.

Statistics may be kept for each individual script which will keep track of information pertaining to that script (or every script if this is used in the parmlib GLOBAL settings). Those parameters are First use date/time, most current use date/time, number of times script was used, and some other statistical parameters. These statistics can be requested from special batch or started task calls to SyzMPF/z.

You can place the STATistics command anywhere within a normal command script, but logically it makes most sense to have it in the system parmlib GLOBAL member, or as the very first parameter of the script.

Parameters:

All	All statistics
Default	Statistics related to script usage, and last use
No	No statistics will be collected

Example:

.....SYS1.PARMLIB(SYZSUB00) member contents...

DSN=SYS1.SyzMPFz.scripts	←Script dataset
ECHO=NONE	←don't echo any commands by default
MODE=NOSIMULATE	←Default setting
MODE=NODEBUG	←Default setting
MODE=NOQUIET	←Default setting
MODE=VERBOSE	←Default setting
STATISTICS=Default	←Default setting
SYZMAIL=Yes	←Allow email to be generated
MAXMSG=100	←Default is 25
To: <u>AllMail@SzygyInc.com</u>	←Default to use if script forgets or is invalid
From: <u>SyzMPFz@SzygyInc.com</u>	←Authorized to send mail from this LPAR
Cc: <u>JOBLOG@SzygyInc.com</u>	←send copy to JOBLOGs
FIRST=@FIRST	←Script DSN member of cmds to do first
LAST=@LAST	←Script DSN member of cmds to do last
.....	

VERBOSE

= YES | No

This global command can still be used but has been deprecated, please use MODE=VERBOSE (defined elsewhere in this manual) The VERBOSE global start up parameter controls the type of messages displayed by SyzMPF/z, YES produces (in some cases) much longer and more detailed messages. This setting is normally not necessary unless asked to be added by Syzygy Support for debugging purposes.

Parameters:

YES	Use the longer and more detailed messages
No	Use standard messages

Example:

.....SYS1.PARMLIB(SYZSUB00) member contents...

DSN=SYS1.SyzMPFz.scripts	←Script dataset
ECHO=NONE	←don't echo any commands by default
MODE=NOSIMULATE	←Default setting
MODE=NODEBUG	←Default setting
MODE=NOQUIET	←Default setting
MODE=VERBOSE	←Default setting
VERBOSE=NO	←Default setting
SYZMAIL=Yes	←Allow email to be generated
MAXMSG=100	←Default is 25
To: AllMail@SyzygyInc.com	←Default to use if script forgets or is invalid
From: SyzMPFz@SyzygyInc.com	←Authorized to send mail from this LPAR
Cc: JOBLOG@SyzygyInc.com	←send copy to JOBLOGs
FIRST=@FIRST	←Script DSN member of cmds to do first
LAST=@LAST	←Script DSN member of cmds to do last
.....	

SIMULATE

This global command has been deprecated, please use `MODE=Simulate`. (defined elsewhere in this manual).

TRACE

= YES | **No**

This global command should only be used if directed by Syzygy Support. It will trace the operation of SyzMPF/z and can/will result in a large number of messages. This setting is normally not necessary unless asked to be added by Syzygy Support for debugging purposes.

Parameters:

YES	Generate trace information between modules
<u>No</u>	Do not trace.

Example:

.....SYS1.PARMLIB(SYZSUB00) member contents...

DSN=SYS1.SyzMPFz.scripts	←Script dataset
ECHO=NONE	←don't echo any commands by default
MODE=NOSIMULATE	←Default setting
MODE=NODEBUG	←Default setting
MODE=NOQUIET	←Default setting
MODE=VERBOSE	←Default setting
TRACE=NO	←Default setting
SYZMAIL=Yes	←Allow email to be generated
MAXMSG=100	←Default is 25
To: AllMail@SyzygyInc.com	←Default to use if script forgets or is invalid
From: SyzMPFz@SyzygyInc.com	←Authorized to send mail from this LPAR
Cc: JOBLOG@SyzygyInc.com	←send copy to JOBLOGs
FIRST=@FIRST	←Script DSN member of cmds to do first
LAST=@LAST	←Script DSN member of cmds to do last
.....	

5. Automatic End of Task Email.

Subject: 00:20:15 SMFDLY(JOB02470) Ended MaxCC=0022 in Step:DAYOMNTH


To: Brian_Westerman@SzygyInc.com
 X-Mailer: SyzMPF/z Version 5.4.0
 Subject: 00:20:15 SMFDLY(JOB02470) Ended MaxCC=0022 in Step:DAYOMNTH

SMFDLY (JOB02470)

Submit via: INTRDR 01/22/2018 at: 00:15:01
 Start: 01/22/2018 at: 00:15:04 Class: R Node: STGE
 Owner: SYZAUTO RacGrp: STCGROUP Msgclass: V
 PgmName: BRIAN WESTERMAN Notify: WESTERB

JobSTEP	ProcSTEP	Program	Status	CCode
NOTTODAY		CRTODAY	Ended	0000
SMFDUMP		SMFDUMP	Ended	0000
RELEASE		ABRDSSU	Ended	0000
DAYOMNTH		DAYOMNTH	Ended	0022
MTHLYNEW		IFASMEDP	Flushed	----
MTHLYOLD		IFASMFDP	Ended	0000
RACFERRS	SAS913	SAS	Ended	0000
EMAIL	XMIT	IKJEFT01	Ended	0000
EMAILERR	XMIT	IKJEFT01	Flushed	----
COMPCODE		SENDCC	Ended	0000
DAYOMNTH		DAYOMNTH	MaxCC	0022

Contact D-Tech support services if you do not want to receive these MaxCC eMail or SMS text messages.

©2014-2017, Syzygy Inc. 

While it is very easy to set up getting eMail from any task that executes in your system, there are several steps to make sure that you handle things correctly. First, in order to use eMail from ANY SyzMPF/z script, you have to make sure that you have the necessary components in place, the first is the standard z/OS SMTP server, which, while it is still available, has been superseded by the z/OS CSSMTP server. The difference between them is that CSSMTP is less prone to failure, and requires far less resources to use. The only advantage that the older SMTP server had was that it allowed receipt of eMail instead of just sending it. Receipt of eMail was a VERY low use part of the product, and its loss was not seen as a problem for most sites.

The second necessary piece is the mechanism that gets the physical eMail package from SyzMPF/z to CSSMTP (or SMTP), and that component is the SyzEMAIL/z add-on. In IBM's infinite wisdom, they decided that message processing and console messages in general should not be allowed to perform any "waits", either direct or inferred. Since all "real i/o" involves an implicit "wait" for that i/o to complete, that means that writing records to any file (even sysout) is strictly out of the question. This add-on provides that connection between SyzMPF/z and CSSMTP.

There are 3 messages which should have SyzMPF/z scripts set for them in order to be sure that all possible tasks are processed under all conditions. The first, and most used MessageID is \$HASP395, which is the message generated on a “normal” task termination. The second MessageID is \$HASP396, which is the message generated when a task ends without even successfully starting, and lastly there is messageID IEFC452I the message that is generated when there was a JCL error before the job was started, (typically a JOBCARD, accounting or job exit failure).

In your Parmlib MPFLSTnn member, you should have the following messages identified to make sure you get all instances of tasks and jobs in your system:

```
$HASP395,SUP(NO),USEREXIT(SYZMPFZ),AUTO(MAXCC) /* Normal End */  
$HASP396,SUP(NO),USEREXIT(SYZMPFZ),AUTO(MAXCC) /* Terminated/JCL */  
IEFC452I,SUP(NO),USEREXIT(SYZMPFZ)           /* JCL Error      */
```

You might have noticed that we have suggested that you use the AUTO parameter in this case for the first two \$HASP** messages. That’s because you can (and should) use the same script for them and to make sure that you don’t end up over time with having to always change two scripts, you can use this method to have them both execute the same exact script.

Processing for the IEFC452I message is kept separate because when a job is terminated in this way, many of the control blocks may not exist. If the JOBCARD itself is invalid, then many of the fields commonly used to make decisions on “who gets” the eMail or text message can be unformatted or corrupted, so it’s best to not even attempt to use them. There will be no steps to this job, so attempting to scan through and get the condition codes for each step would be meaningless and the output would be quite useless. But... Someone does need to know that this happened, and there will always be a taskname (although since the JOBCARD was damaged in some way, it might not be useful, but generally, the taskname will be okay to use.

Sample script for Error processing

IEFC452I Script

```
*012345678901234567890123456789012345678901234567890  
*IEFC452I taskname JCLERR
```

```
*
*ECHO=ALL <- only needed for testing
IF TASKNAME = PAY* | HR* ++ IF TASKNAME ^= PAYT*
  CC: Payroll@mycompany.com
  CC: Chester@mycompany.com
ENDIF
IF TASKNAME = APP* | JRP* | JIP* | JPP* | PSP*
  CC: Acct-ApplicationsS@mycompany.net
  CC: Opers
ENDIF
IFTASKNAME = CAP* | GAP* | P%P* | TXP* | RDP*
  CC: General-Applications@mycompany.com
  CC: Opers
ENDIF
IF TASKNAME = ADA* | COMP* | BROK* | EXX*
  CC: ADABAS
ENDIF
EMAIL
SUBJect: &HH:&MM:&SS &TASKNAME(&TASKID) - Not Run -, JCL ERROR
TO: Brian_Westerman@SyzygyInc.com
FROM: Prod_errors@MyCompany.com
CC: BTEXT
MSG: &TASKNAME(&TASKID) was not able to execute due to a JCL Error
SENDMAIL      ←this is the command to actually send the email/text
```

The above will always send an email to Brian_Westerman@SyzygyInc.com with the time of the error, (the date is already assumed because of the fact that it's an email, and a text is sent to Brian as well via the "nickname" BTEXT. Additionally, depending on the task name additional "CC:" are added. While this may seem like a lot of work, it actually is processed totally in memory and the overhead is not measurable, (at least we have not been able to measure it) to greater than .000 (3 decimal points) of time.

It should be noted that there is no attempt to process step condition codes (which would have included "EMAIL MAXCC STEPCC" instead of simply "EMAIL". The longer form of the EMAIL command tells SyzMPF/z to process all of the steps of the task, and to provide the Maximum condition code as well in a formatted email. The shorter form merely states that an email or TEXT is generated and it may (or may not) contain any HTML formatting. In this case, it does not contain any HTML formatting.

Sample script for regular task processing

MAXCC script for \$HASP395 & \$HASP396

Normal task ends in our example (and abnormal ones) are handled with the same script (MAXCC) which is shown below:

```
*0123456789012345678901234567890123456789012345678901234567890
*$HASP395 taskname ENDED
*
*ECHO=ALL <- only needed for testing
IF MSGTYPE TSU           <- skip if this is a TSO user logging off
  (EXIT)
ENDIF
IFMAXCC > 0              <-If the MaxCC is higher than zero
  ATTACH=JESMSGLOG       <- Attach the JOB and SYSTEM message logs
  ATTACH=JESSYMSG        to this output
ENDIF
TO: Brian Westerman      <- in case all else fails, tell Brian about this
FROM: Production@MyCompany.com <- in case we forgot the parmlib default(s)
*WTO &TASKNAME &TASKID &MAXCC
IFNOTIFY EQ RON*         <- if one of RON's jobs, send email no matter what
  EMAIL MAXCC STEPCC
  TO: RON
  SENDMAIL               <- send the mail,
  (EXIT)                 <-and exit
ENDIF
*Now we do the conditional emails (only "if" something)
IFMAXCC > 0              <-If MAXCC greater than zero
  EMAIL MAXCC STEPCC    <- Generate all Step codes and MAX
  IFTASKNAME = SYZ*
  TO: BRIAN
ENDIF
IF NOTIFY = BRIAN | JOHN | PEPPER | A* | H*
  TO: &NOTIFY           <-Send it to whoever the notify said gets it
ENDIF
IFTASKNAME = HR* | PAY*
  TO: Payroll@mycompany.com
ENDIF
SENDMAIL
(EXIT)
ENDIF
*IF MAXCC = 0
*do nothing
```

*endif

In the above MAXCC script, we first make sure that we aren't just looking at a TSO user ended message, because (at least for us) no one cares about being notified via eMail or test message that they logged off.

It's also a good idea to have set up a default to get messages in case someone generates a script the ends up with an eMail being sent, but no one is set in the "TO:" field. Actually, we suggest that you do this in the parmlib startup parms (specifying a default TO:, FROM: etc.).

Next you can set up your script to check for any fields of the task that you want, taskname, notify, job owner/submitter, RACF group, (it's a long list of choices), and you can specify based on that check, who is to get the eMail or text message. You can use actual email addresses, or you can use nicknames, which can be from 1 to 256 recipients or text addresses to receive the email (see the SyzMAIL/z documentation for how to use nicknames). When a recipient is named that does NOT include an ampersand "@", it's assumed that it refers to a "nickname" and no extra checking is performed at that time.

Processing continues until there is no more script, or an (EXIT) command is reached. There is no limit to the length of the script, well, there is but it's over 250,000 lines. You can also make things easier to maintain if you use SKIPTO to jump to other scripts to do common formatting such as the processing for abends or doing checking just before sending. Some sites create a script called "SENDIT" and then use the main script to do all of the normal IF processing, then at the end, they SKIPT SENDIT, where it does some sanity checking on the results like making sure it actually is going to go to "someone" or that if it's a TEXT, maybe they don't want the 500 line JES messages or if it's at night, make sure that if no one were being texted, that at least someone is operations gets a text message that something bad happened. Some sites also use that ending script to add a disclaimer to their emails.

There are no strict "everybody does this" type of scripts. In the more than 300 sites (as of 2017) that have licensed the products, I don't think any two of them have anything approaching identical condition code processing scripts.

If you have any problems generating a script that fits your needs, feel free to contact Syzygy (800) 767-2244 and let us help you set it up. Once you get started, you will see that it is actually quite simple. Most sites send email for everything at the beginning and then taper off a bit after a while. It seems like a good idea (at first) to be notified for every job that runs, but the novelty soon wears off and many people ask to only get the notify if the job gets something "other" than a MAXCC of ZERO.

6. Control Commands:

*** (Comment)**

The * control command is used to allow comments to be entered in the command script. These lines are not executed in any way. There is no limit to the number of comments in the script, and there is no limit to the contents of the comment.

Parameters:

NONE

Example:

```
.....
* This is an example of a comment
***** As long as we begin the comment with an asterisk in column 1 we are okay
*****
*   Comment                                     *
*****
.....
```

&REPLYID

The &REPLYID control command is actually not a command at all, but is the representation of the REPLY ID of the WTOR message that was saved in a variable by the "GET REPLYID" command (Discussed later in this manual). This variable can be used anywhere within any line of the script, and the "&REPLYID" is replaced by the contents of the saved variable "&REPLYID". You could use the REPLY command to perform this same type of function, but there are times when you might need to have more flexibility.

Parameters:

NONE

Example: 00 01 02 03 04 05 06 07
 @31 ABC123I is this a system level UPDATE ?

.....
*

GET REPLYID (The REPLYID of the WTOR we are processing is saved)
GET WORD 06 (The word we captured into the variable is the word "UPDATE")
..... lots of other commands

R &REPLYID,&WORD

"results in **R 31,UPDATE** "

.....

&STR (or &STRING)

The &STR control command is actually not a command at all, but is the representation of the STRING of the message that was saved in a variable by the "GET STRING" command (Discussed later in this manual). This variable can be used anywhere within any line of the script, and the "&STR" (or &STRING) is replaced by the contents of the saved variable "&STR". &STR and &STRING can be used interchangeably.

Parameters:

NONE

```

                                1           2           3           4
01234567890123456789012345678901234567890
```

Example: ABC12345 This is a test system UPDATE

.....

*

GET STRING 31(06) (The word we captured into the variable is the word "UPDATE")

..... lots of other commands

S TEST,M=&STRING (Results in "S TEST,M=**UPDATE**")

.....

&variables - STATIC Variables

The *&variable* control command is technically not a command at all, but is the representation of a preset STATIC variable which will replace the *&variable* keyword with the actual value of that variable. There are currently over 30 static variables that can be used in this way. These STATIC variables can be inserted within character strings and will be processed "on the fly" and replace the variable with the number of characters represented by that variable. For instance, if you wanted to include the current time in a WTO message, you could state "WTO=The current time is &HH:&MM:&SS" which will replace the three STATIC variables &HH, &MM and &SS so that the actual command that is process is "WTO The current time is 11:23:55" (Assuming it was really 11:23:55).

Parameters:

NONE REQUIRED

TIME Related STATIC variables:

&HH - Hour of the current Time of Day (24 Hour Clock)
&MM - Minutes of current Time of Day (24 Hour Clock)
&SS - Seconds of the current Time of Day (24 Hour Clock)
&HHMM - Current Hours and Minutes (24 Hour Clock)
&HHMMSS - Current Hours, Minutes and Seconds (24 Hour Clock)

Example:

.....
WTO The current time is &HH:&MM&SS
**** Results in the following WTO:
"The current time is **20:35:58**"

.....

DATE Related STATIC variables:

&MN - (not MM) Current Month digits
&DD - Current Day digits
&YY - Current Year digits
&MNDD or MMDD - Current Month and Day digits
&MNDDYY or MMDDYY - Current Month, Day and Year digits

Example:

.....
WTO The current Date is &MN/DD/YY ←-note Month is &MN when single not &MM
**** Results in the following WTO:
"The current date is 11/23/12"

.....

&ASID - Address space ID STATIC variable:

&ASID - 4 character Address space ID (in displayable Hex)

Example:

```
.....  
WTO The current ASID is &ASID  
**** Results in the following WTO:  
"The current ASID is 001A"
```

```
.....
```

&BCC: - Email Blind Carbon Copy (":" is required) STATIC variable:

&CC: - 1 to 60 character Blind Carbon Copy ID currently set

Example:

```
.....  
WTO The current BCC: is &BCC:  
**** Results in the following WTO:  
"The current BCC: is Fred@Flintstone.com"  
.....
```


&CC: - Email Carbon Copy (":" is required) STATIC variable:

&CC: - 1 to 60 character Carbon Copy ID currently set

Example:

```
.....  
WTO The current CC: is &CC:  
**** Results in the following WTO:  
"The current CC: is Fred@Flintstone.com"  
.....
```

&CONID - Console ID STATIC variable:

&CONID - 1 to 4 character CONSOLE ID that issued this command
 (If not a command this will be meaningless)
 (Data is displayed as 8 hex characters)

Example:

.....

WTO The current CONSOLE ID is &CONID

**** Results in the following WTO:

"The current CONSOLEID is 00000002" <- Console ID is '2'

.....

&CONSOLE - Console name STATIC variable:

&CONSOLE - 1 to 8 character CONSOLE name that issued this message or command (If this is a message and not a command then the console name can be meaningless.), it can be INTERNAL, INTRDR, as TSOUSER or the actual console name if command came from an operator console

Example:

.....

WTO The current CONSOLE name is &CONSOLE

**** Results in the following WTO:

"The current CONSOLE name is **BETATST**" ←- BETATST is a TSO user who issued the CMD

.....

&CPUID - CPU Serial Number STATIC variable:

&CPUID - 4 character CPU Serial (does not include LPAR# portion)

Example:

```
.....  
WTO The current CPU Serial number is &CPUID  
**** Results in the following WTO:  
"The current CPU Serial number is 1234"  
  
.....
```

&CPUTYPE - Mainframe Model Type STATIC variable:

&CPUTYPE - 4 character Hardware Model type

Example:

.....
WTO The current Hardware model type is &CPUTYPE
**** Results in the following WTO:
"The current Hardware model type is 2086"
.....

&DOW - Day of Week STATIC variable:

&DOW - Three Character Day of Week (MON-TUE-WED-THU-FRI-SAT-SUN)

Example:

.....
WTO The current DOW is &DOW
**** Results in the following WTO:
"The current DOW is **MON**"

.....

&EXECCLASS- Execution CLASS of this task

&execclass - 1 character task Execution class

Example:

```
.....  
WTO The current Execution CLASS is &EXECCLASS  
**** Results in the following WTO:  
"The current Execution CLASS is A"  
  
.....
```

&From: - Email From (":" is required) STATIC variable:

&FROM: - 1 to 60 character FROM ID currently set

Example:

.....
WTO The current FROM: is &FROM:
**** Results in the following WTO:
"The current FROM: is Mickey@Mouse.com"
.....

&LPAR - Logical Partition Name STATIC variable:

&LPAR - 1 to 8 character Logical Partition Name

Example:

.....
WTO The current LPAR is &LPAR
**** Results in the following WTO:
"The current LPAR is **SYSTEMA**"

.....

&MaxCC - Maximum Condition code (so far):

&maxcc - 4 character maximum condition code (or abend code)

Example:

```
.....  
WTO The current MaxCC (so far) is &MAXCC  
**** Results in the following WTO:  
"The current MaxCC (so far) is 0123"  
  
.....
```

&MAXPROGRAM- Step Program with the Maximum condition code (so far):

&maxprogram - 1 - 8 character step program name with the maximum condition code (or
abend code)

Example:

.....
WTO The current MaxPROGRAM(so far) is &MAXprogram
**** Results in the following WTO:
"The current MaxPROGRAM (so far) is **IEBGENER**"

.....

&MAXPSTEP- ProcStep Name with the Maximum condition code (so far):

&maxpstep - 1 - 8 character proc-step name with the maximum condition code (or abend code)

Example:

.....
WTO The current MaxPSTEP(so far) is &MAXpstep
**** Results in the following WTO:
"The current MaxPSTEP (so far) is **Pstep08A**"

.....

&MAXRESULT- Highest Task Result

&maxresult - 1 - 8 character task result, normally ENDED, FLUSHED, JCLERROR, ABENDED, NOTRUN.

Example:

.....

WTO The current MaxRESULT(so far) is &MAXresult

**** Results in the following WTO:

"The current MaxRESULT (so far) is **ABENDED**"

.....

&MAXSTEP- Step with the Maximum condition code (so far):

&maxstep - 1 - 8 character step name with the maximum condition code (or abend code)

Example:

.....

WTO The current MaxSTEP(so far) is &MAXstep

**** Results in the following WTO:

"The current MaxSTEP (so far) is **STEP012A**"

.....

&MON - Month STATIC variable:

&MON - Three Character MONTH (JAN-DEC)

Example:

```
.....  
WTO The current Month is &MON  
**** Results in the following WTO:  
"The current Month is NOV"  
  
.....
```

&MSGCLASS- MSGCLASS of this task:

&msgclass - 1 character task MSGCLASS

Example:

```
.....  
WTO The current MSGCLASS is &MSGCLASS  
**** Results in the following WTO:  
"The current MSGCLASS is A"  
  
.....
```


&MSGID - Current Message being processed STATIC variable:

&MSGID - 1 to 48 character Message number was are processing
 (This is actually the first (up to) 48 characters of the message line
 that caused us to be processing this script)

Example:

.....
WTO The current Message being processed is &MSGID
**** Results in the following WTO:
"The current Message being processed is **IEA141I**"

.....

&MSGTEXT - Current Message being processed - STATIC variable:

&MSGTEXT - first 126 characters of the Message we are processing
(This is actually the first (up to) 126 characters of the message line
that caused us to be processing this script).
*Note, if you don't have space, the command line will be truncated

Example:

.....
WTO The current Message being processed is &MSGTEXT
**** Results in the following WTO:
"The current Message being processed is **\$HASP395 SYZMPF54 ENDED - RC=0000** "
.....

&NOTIFY- JOB card or /* JECL NOTIFY= :

&NOTIFY - 1 – 8 character NOTIFY= parm name

Example:

.....
WTO The current tasks NOTIFY=¬ify
**** Results in the following WTO:
"The current tasks NOTIFY=**BRIANW**"

.....

&ORIGIN - Origin of this task (System name) STATIC variable:

&ORIGIN - 4 character Subsystem this task started on (Originally)

Example:

```
.....  
WTO The current ORIGIN is &ORIGIN  
**** Results in the following WTO:  
"The current ORIGIN is JESA"  
  
.....
```

&PGMRNAME- JOBCARD Programmer Name of this task:

&pgmrname - 1 - 20 character JOB Programmername

Example:

.....
WTO The current task has a Programmer name of &PGMRNAME
**** Results in the following WTO:
"The current task has a Programmer name of **Payroll Job 1 of 2**"

.....

&SECGROUP | &RACFGRP - Security group of this task:

&SECGROUP - 1 – 8 character task RACF group this task belongs to

Example:

.....

WTO The current RACF group is &racfgrp

**** Results in the following WTO:

"The current RACF group is **SYSSTC**"

.....

&SCRIPTID - Currently executing Script name STATIC variable:

&SCRIPT - 1 to 8 character Script name that is currently executing this command

Example:

```
.....  
WTO The current Script Name is &SCRIPTID  
**** Results in the following WTO:  
"The current Script name is IEE141I" ← Member name in the Script PDS  
  
.....
```

&SYSID - System ID STATIC variable:

&SYSID - 1 to 8 character SYSTEM ID

Example:

.....

WTO The current SYSTEM Identification name is &SYSID

**** Results in the following WTO:

"The current SYSTEM Identification name is **SYSA**"

.....

&SYSPLEX - SysPLEX Name STATIC variable:

&SYSPLEX - 1 to 8 character SYSPLEX Name

Example:

```
.....  
WTO The current SYSPLEX Name &SYSPLEX  
**** Results in the following WTO:  
"The current SYSPLEX Name is SYZYGYPL"  
  
.....
```

&SYSPLEXID - SysPLEX ID STATIC variable:

&SYSPLEXID - 2 character SYSPLEX ID

Example:

```
.....  
WTO The current SYSPLEX ID &SYSPLEXID  
**** Results in the following WTO:  
"The current SYSPLEX ID is 0A"  
  
.....
```

&STARTDATE- Date this task began execution

&startdate - 8 character date mm/dd/yy

Example:

.....

WTO This task started on &STARTDATE at &STARTTIME

**** Results in the following WTO:

"This task started on **11/23/2013 at 11:22:18**"

.....

&STARTTIME - **Time this task began execution**

&starttime - 8 character time hh:mm:ss

Example:

.....

WTO This task started on &STARTDATE at &STARTTIME

**** Results in the following WTO:

“This task started on **11/23/2013 at 11:22:18**”

.....

&SUBDATE- Date this task entered (or submitted to) the system:

&subdate - 8 character date mm/dd/yy

Example:

.....

WTO This task was submitted on &SUBDATE at &SUBTIME

**** Results in the following WTO:

“This task was submitted on **11/23/2013 at 11:22:16**”

.....

&SUBJECT: - Email SUBJECT (":" is required) STATIC variable:

&Subject: - 1 to 60 character Email Subject currently set

Example:

.....
WTO The current email subject: is &SUBJECT:
**** Results in the following WTO:
"The current email subject: is How to eat candy"
.....

&SUBMETH - Submit Method (resource) of this task:

&submeth - 1 – 8 Submit resource name

Example:

.....

WTO This task was submitted via &SUBMETH

**** Results in the following WTO:

"This task was submitted via **INTRDR**"

.....

&SUBTIME- Time this task entered (or submitted to) the system:

&subtime - 8 character time hh:mm:ss

Example:

.....
WTO This task was submitted on &SUBDATE at &SUBTIME
**** Results in the following WTO:
"This task was submitted on **11/23/2013 at 11:22:16**"

.....

&TASKNAME - Task Name STATIC variable:

&TASKNAME - 1 to 8 character TASK Name (i.e jobname)

Example:

```
.....  
WTO The current Task Name is &TASKNAME  
**** Results in the following WTO:  
"The current Task Name is CICS001"  
  
.....
```

&TASKID - TaskID (TASK-number) ID STATIC variable:

&TASKID - 8 character TASK Number (JOBnnnnn, STCnnnnn, TSUnnnnn)

Example:

```
.....  
WTO The current JOB Number is &TASKID  
**** Results in the following WTO:  
"The current JOB Number is STC01233"  
  
.....
```

&TASKTYPE - Task Type STATIC variable:

&TASKTYPE - 3 character TASK Type (JOB, STC, TSU)

Example:

```
.....  
WTO The current Task Type is &TASKTYPE  
**** Results in the following WTO:  
"The current Task Type is STC"  
  
.....
```

&TO: - Email TO id (":" is required) STATIC variable:

&TO: - 1 to 60 character email TO ID currently set

Example:

.....
WTO The current email to is &TO:
**** Results in the following WTO:
"The current email to: is Wilma@Flintstone.com"
.....

&USERID | &TASKOWNER - Task Owner, submitter or USER= of this job :

&USERID - 1 – 8 character USER= name or RACF/ACF2 owner of this task

Example:

.....
WTO The current tasks OWNER=&userid
**** Results in the following WTO:
"The current tasks OWNER=**BRIAN**"

.....

&VERSION - Display the SyzMPF/z version, assembly date and time

&VERSION - displays 8character version, 8 character date and 5 character time of Product

Example:

.....

WTO The SyzMPF/z version is "&VERSION"

**** Results in the following WTO:

"The SyzMPF/z version is "**05.07.01 11/23/2017 23.01**"

.....

&<variable> - use previously set user variable:

&<variable> - 1 to 12 character variable (named within < > carrots which will contain the
1 to 16 character actual variable that was previously set.

Example:

```
.....  
SETVAR <TEST_VARIABL> I am a TEST Var  
.....  
WTO The value of <TEST_VARIABL> is &<TEST_VARIABL>  
**** Results in the following WTO:  
"The value of <TEST_VARIABLE> is I am a TEST VaR"  
.....
```

&WORKLOAD- WLM WORKLOAD name of this task:

&workload - 1 – 8 character WLM workload name

Example:

.....

WTO The current WLM workload for this task is &WORKLOAD

**** Results in the following WTO:

"The current WLM workload for this task is BATHIGH"

.....

DECVAR or DECVART

nn

Note: Numeric Variables are set via SETVAR or SETVART commands.

The DECVAR and DECVART (if it's a non-persistent or temporary variable) control commands are used to cause SyzMPF/z to decrement (reduce a variable) a numeric variable by some number (the default is '1' (one). The variable can be decremented to zero before it will send a message that it cannot be decremented any further.

The variable being used MUST be NUMERIC, or it will generate an error and no decrement process will occur for that variable.

Parameters:

nn

The number to decrement the variable by

Example:

.....For message

WTO Testing Increment & decrement of a variable

IF<#TEST1> \$ON\$

ELSE

SETVAR <#TEST1> 4 ←start with Variable set to "4"

ENDIF

WTO Variable #TEST1 is set to &<#TEST1>

DECVAR <#TEST1> 2 ← Variable is now set to '2' (4-2=2)

WTO Variable #TEST1 is NOW &<#TEST1>

INCVAR <#TEST1> 1 ← variable is now set to '3' (2+1=3)

WTO Variable #TEST1 is NOW &<#TEST1>

.....

INCVAR or INCVART

nn

Note: Numeric Variables are set via SETVAR or SETVART commands.

The INCVAR and INCVART (if it's a non-persistent or temporary variable) control commands are used to cause SyzMPF/z to increment a numeric variable by some number (the default is '1' (one)). The variable can be incremented up to 9,999,999,999 (just under 10 billion) before it will overflow.

The variable being used MUST be NUMERIC, or it will generate an error and no increment process will occur for that variable.

Parameters:

nn

The number to increment the variable by

Example:

.....For message

WTO Testing Increment & decrement of a variable

IF<#TEST1> \$ON\$

ELSE

SETVAR <#TEST1> 4 ←start with Variable set to "4"

ENDIF

WTO Variable #TEST1 is set to &<#TEST1>

DECVAR <#TEST1> 2 ← Variable is now set to '2' (4-2=2)

WTO Variable #TEST1 is NOW &<#TEST1>

INCVAR <#TEST1> 1 ← variable is now set to '3' (2+1=3)

WTO Variable #TEST1 is NOW &<#TEST1>

.....

SETVAR - Setting Persistent Variables

<Variable name> 0123456789abcdef (up to 16 characters or spaces)
 \$\$BLANK\$\$\$ (sets variable to blanks)

“SETVARP” is an alias for SETVAR

“SETPVAR” is an alias for SETVAR

The SETVAR control command the way to create Persistent variables within the scripts. Persistent variables will exist beyond the length of the task that caused the script to execute which created the variable. The variable will continue to exist for the length of the IPL or until specifically deleted via the DELVAR subcommand. They may also be replaced via the REPVAR command, both are described later in this document.

The variable name “<variable name>” can be made up from any ebcdic supported characters, upper and lower case up to 12 bytes in length not including the “<>” (greater than/less than) characters. Spaces within the name are supported. i.e. <TST_VARIABLE> (12 bytes) is preferable to <TST VARIABLE>, but both are supported. Also note that <Test_Var> and <TEST_VAR> are **completely** different variables because of the upper/lower case character differences.

The actual variable contents can be anything at all, but is limited to 16 characters including any spaces or special characters. Any character can be used including non-printable hex characters. Other variables and STATIC variables can be used as well in both the <variable name> (variable name) and the actual contents of the variable.

Example: If you were processing messageID IEF403I for CICS001:

```
IEF403I CICS001 - STARTED - TIME=20.06.17
W00   W01       W03       W05
```

If you were to specify “SETVAR <MY_&W01> ST &W05 data”

SyzMPF/z will create a variable named “MY_CICS001”
Which will contain: “**ST TIME=20.06.17**”

If you were to specify more than the allowed 16 characters for the contents of the variable, they will be truncated at 16 bytes and a warning message will be issued.

Persistent variables can be created, deleted or replaced via supported commands described in this manual. There can be (almost) any number of spaces between the <variable name> name and the actual contents of the variable, but please note that

the contents of the variable will begin with the **first NON-Blank** character following the *<variable name>*.

Variables can also be tested and compared with the supported "IF *<variable name>*" script command. They can be compared for actual contents, or just tested for existence, regardless of the contents.

You will receive an error message under any of these instances and a return code is set which can be tested by your processing, variable errors are not considered fatal:

You attempt to create a variable that already exists

You attempt to REPLACE the contents of a variable that does not yet exist.

If you attempt to DELETE a variable that does not yet exist, you will receive an error, but it is informational only.

Possible Return codes from sub-commands:

DELETE:

VRC=0	OK
VRC=4	Delete Failure
VRC=9	Command failure (user error, see messages)

SET/CREATE:

VRC=0	OK
VRC=2	Created with blanks (no actual variable found)
VRC=4	Variable already existed, replaced with new value
VRC=8	Create failure
VRC=9	Command failure (user error, see messages)

REPLACE:

VRC=0	OK
VRC=2	Variable was replaced with BLANKS, no actual value provided
VRC=6	Variable no longer viable (probable z/OS system error) (deleted)
VRC=7	Variable no longer viable (probable z/OS system error) (still exists)
VRC=8	Variable did not already Exist, not created
VRC=9	Command failure (user error, see messages)

Parameters:

up to 16 byte variable contents (begins with first non-blank)

Example:

```
.....
SETVAR <BRI-10-123> 0123456789ABCDEF
SETVAR <BRI-10-9AB> ABCDEFGHIJKLMNOPQRS <truncates at "P"
SETVAR <CICSisUP> &MM/DD-&HH:MM:SS (results in 11/23-01:21:13)

...
WTO CICS started at &<CICSisUP>
***** this sends a WTO stating "CICS started at 11/23-01:21:13" to the console

...
** if CICS is up is set then start something, $ON$ just checks that it's set to anything
** $OFF$ checks if it not set to anything, i.e. does not exist.
IF <CICSisUP> = $ON$
    Go do something
ELSE
    Go do something else
ENDIF
....
```

SETVART - Setting NON-Persistent (temporary) Variables

<Variable name> 0123456789abcdef (up to 16 characters or spaces)
 \$\$BLANK\$\$ (sets variable to blanks)

“SETTVAR” is an alias for SETVART

“SETVARNP” is an alias for SETVART

The SETVART control command the way to create NON-Persistent variables within the scripts. NON-Persistent variables will exist only for the length of the task that caused the script to execute which created the variable. For instance, if the message that contained the SETVART command in the script was generated by CICS001, then any NON-Persistent variable created with the SETVART command within that script will cease to exist when CICS001 ends.

There is no time limit, but when the address space ends so does the variable's existence. The variable will be automatically removed when the task that created the variable (which is the one that issued the message that SyzMPF/z is processing) ends.

Care should be taken to make sure that you don't logically mix up the Persistent and NON-Persistent variables because the only difference between them is how long they exist.

The variable will continue to exist for the length of the task that created the message whose script was used to create the variable, after which it will be automatically removed. Also the variable can be specifically deleted via the DELVAR (or DELVART) subcommand. They may also be replaced via the REPVART command, both are described later in this document.

The site may decide at any time to change the variable from NON-Persistent to Persistent simply by using the " REPVAR *<variable name>* &*<variable name>* " as described later in this document.

The variable name "*<variable name>*" can be made up from any ebcdic supported characters, upper and lower case up to 12 bytes in length not including the "<>" (greater than/less than) characters. Spaces within the name are supported. i.e. <TST_VARIABLE> (12 bytes) is preferable to <TST VARIABLE>, but both are supported. Also note that <Test_Var> and <TEST_VAR> are **completely** different variables because of the upper/lower case character differences.

The actual variable contents can be anything at all, but is limited to 16 characters including any spaces or special characters. Any character can be used including non-

printable hex characters. Other variables and STATIC variables can be used as well in both the *<variable name>* (variable name) and the actual contents of the variable.

Example: If you were processing messageID IEF403I for CICS001:

```
IEF403I CICS001 - STARTED - TIME=20.06.17
  W00   W01       W03       W05
```

If you were to specify "SETVART <MY_&W01> ST &W05 data"

SyzMPF/z will create a variable named "MY_CICS001"
which will contain: "ST TIME=20.06.17"

If you were to specify more than the allowed 16 characters for the contents of the variable, they will be truncated at 16 bytes and a warning message will be issued.

NON-Persistent variables can be created, deleted or replaced via supported commands described in this manual. There can be (almost) any number of spaces between the *<variable name>* name and the actual contents of the variable, but please note that the contents of the variable will begin with the **first NON-Blank** character following the *<variable name>*.

Variables can also be tested and compared with the supported "IF *<variable name>*" script command. They can be compared for actual contents, or just tested for existence, regardless of the contents.

You will receive an error message under any of these instances and a return code is set which can be tested by your processing, variable errors are not considered fatal:

You attempt to create a variable that already exists

You attempt to REPLACE the contents of a variable that does not yet exist.

If you attempt to DELETE a variable that does not yet exist, you will receive an error, but it is informational only.

Possible Return codes from sub-commands:

DELETE:

VRC=0	OK
VRC=4	Delete Failure
VRC=9	Command failure (user error, see messages)

SET/CREATE:

VRC=0	OK
VRC=2	Created with blanks (no actual variable found)
VRC=4	Variable already existed, replaced with new value
VRC=8	Create failure
VRC=9	Command failure (user error, see messages)

REPLACE:

VRC=0	OK
VRC=2	Variable was replaced with BLANKS, no actual value provided
VRC=6	Variable no longer viable (probable z/OS system error) (deleted)
VRC=7	Variable no longer viable (probable z/OS system error) (still exists)
VRC=8	Variable did not already Exist, not created
VRC=9	Command failure (user error, see messages)

Parameters:

up to 16 byte variable contents (begins with first non-blank)

Example:

```
.....
SETVART <BRI-10-123> 0123456789ABCDEF
SETTVAR <BRI-10-9AB>  ABCDEFGHIJKLMNOPQRS <truncates at "P"
SETVARNP <CICSisUP>  &MM/DD-&HH:MM:SS (results in 11/23-01:21:13)

...
WTO CICS started at &<CICSisUP>
***** this sends a WTO stating "CICS started at 11/23-01:21:13" to the console

...
** if CICS is up is set then start something, $ON$ just checks that it's set to anything
**  $OFF$ checks if it not set to anything, i.e. does not exist.
IF <CICSisUP> =  $ON$
    Go do something
ELSE
    Go do something else
ENDIF
....
```

REPVAR - Replacing Persistent Variables

<Variable name> 0123456789abcdef (up to 16 characters or spaces)
 \$\$BLANK\$\$ (sets variable to blanks)

“REPVARP” is an alias for REPVAR

“REPPAR” is an alias for REPVAR

The REPVAR control command the way to replace already existing Persistent variables within the scripts. Persistent variables will exist beyond the length of the task that caused the script to execute which created the variable. The variable will continue to exist for the length of the IPL or until specifically deleted via the DELVAR subcommand. You can also use the REPVAR command to change a NON-Persistent variable to Persistent.

Example: Change the NON-persistent variable called <MyVar> to persistent
 REPVAR <MyVar> &<MyVar>

The variable name “<variable name>” can be made up from any ebcdic supported characters, upper and lower case, up to 12 bytes in length not including the “<>” (greater than/less than) characters. Spaces within the name are supported. i.e. <TST_VARIABLE> (12 bytes) is preferable to <TST VARIABLE>, but both are supported. Also note that <Test_Var> and <TEST_VAR> are **completely** different variables because of the upper/lower case character differences.

The actual variable contents can be anything at all, but is limited to 16 characters including any spaces or special characters. Any character can be used including non-printable hex characters. Other variables and STATIC variables can be used as well in both the <variable name> (variable name) and the actual contents of the variable.

Example: If you were processing messageID IEF403I for CICS001:

```
IEF403I CICS001 - STARTED - TIME=20.06.17
      W00   W01       W03       W05
```

If you were to specify “REPVAR <MY_&W01> ST &W05 data”

SyzMPF/z will check for a variable named **“MY_CICS001”**
And if found will replace the contents with: **“ST TIME=20.06.17”**

If you were to specify more than the allowed 16 characters for the contents of the variable, they will be truncated at 16 bytes and a warning message will be issued.

There can be (almost) any number of spaces between the <variable name> name and the actual replacement contents of the variable, but please note that the contents of the variable will begin with the **first NON-Blank** character following the <variable name>.

Variables can also be tested and compared with the supported "IF <variable name>" script command. They can be compared for actual contents, or just tested for existence, regardless of the contents.

You will receive an error message under any of these instances and a return code is set which can be tested by your processing, variable errors are not considered fatal:

You attempt to create a variable that already exists

You attempt to REPLACE the contents of a variable that does not yet exist.

If you attempt to DELETE a variable that does not yet exist, you will receive an error, but it is informational only.

Possible Return codes from sub-commands:

DELETE:

VRC=0	OK
VRC=4	Delete Failure
VRC=9	Command failure (user error, see messages)

SET/CREATE:

VRC=0	OK
VRC=2	Created with blanks (no actual variable found)
VRC=4	Variable already existed, replaced with new value
VRC=8	Create failure
VRC=9	Command failure (user error, see messages)

REPLACE:

VRC=0	OK
VRC=2	Variable was replaced with BLANKS, no actual value provided
VRC=6	Variable no longer viable (probable z/OS system error) (deleted)
VRC=7	Variable no longer viable (probable z/OS system error) (sill exists)
VRC=8	Variable did not already Exist, not created
VRC=9	Command failure (user error, see messages)

Parameters:

up to 16 byte variable contents (begins with first non-blank)

Example:

```
.....
REPVAR <BRI-10-123> 0123456789ABCDEF
REPVARP <BRI-10-9AB> ABCDEFGHIJKLMNOPQRS <truncates at "P"
REPVAR <CICSisUP> &MM/DD-&HH:MM:SS (results in 11/23-01:21:13)

...
WTO CICS started at &<CICSisUP>
***** this sends a WTO stating "CICS started at 11/23-01:21:13" to the console

...
** if CICS is up is set then start something, $ON$ just checks that it's set to anything
** $OFF$ checks if it not set to anything, i.e. does not exist.
IF <CICSisUP> = $ON$
    Go do something
ELSE
    Go do something else
ENDIF
....
```

REPVART - Replacing NON-Persistent Variables

<Variable name> 0123456789abcdef (up to 16 characters or spaces)
 \$\$BLANK\$\$ (sets variable to blanks)

“REPTVAR” is an alias for REPVART

“REPVARNP” is an alias for REPVART

The REPVART control command the way to replace already existing NON-Persistent variables within the scripts. NON-Persistent variables will exist only for the length of the task that caused the script to execute which created the variable. For instance, if the message that contained the SETVART command in the script was generated by CICS001, then any NON-Persistent variable created with the SETVART command within that script will cease to exist when CICS001 ends.

There is no time limit, but when the address space ends so does the variable's existence. The variable will be automatically removed when the task that created the variable (which is the one that issued the message that SyzMPF/z is processing) ends.

Care should be taken to make sure that you don't logically mix up the Persistent and NON-Persistent variables because the only difference between them is how long they exist.

The variable will continue to exist for the length of the task that created the message whose script was used to create the variable, after which it will be automatically removed. Also the variable can be specifically deleted via the DELVAR (or DELVART) subcommand.

The site may decide at any time to change the variable from NON-Persistent to Persistent simply by using the " REPVAR *<variable name>* &*<variable name>* " as described later in this document.

Example: Change the NON-persistent variable called *<MyVar>* to persistent
 REPVAR *<MyVar>* &*<MyVar>*

The variable name "*<variable name>*" can be made up from any ebcdic supported characters, upper and lower case, up to 12 bytes in length not including the "<>" (greater than/less than) characters. Spaces within the name are supported. i.e. *<TST_VARIABLE>* (12 bytes) is preferable to *<TST VARIABLE>*, but both are supported. Also note that *<Test_Var>* and *<TEST_VAR>* are **completely** different variables because of the upper/lower case character differences.

The actual variable contents can be anything at all, but is limited to 16 characters including any spaces or special characters. Any character can be used including non-printable hex characters. Other variables and STATIC variables can be used as well in both the *<variable name>* (variable name) and the actual contents of the variable.

Example: If you were processing messageID IEF403I for CICS001:

```
IEF403I CICS001 - STARTED - TIME=20.06.17
W00  W01  W03  W05
```

If you were to specify "REPVART <MY_&W01> ST &W05 data"

SyzMPF/z will check for a variable named "MY_CICS001"
And if found will replace the contents with: "ST TIME=20.06.17"

If you were to specify more than the allowed 16 characters for the contents of the variable, they will be truncated at 16 bytes and a warning message will be issued.

There can be (almost) any number of spaces between the *<variable name>* name and the actual replacement contents of the variable, but please note that the contents of the variable will begin with the **first NON-Blank** character following the *<variable name>*.

Variables can also be tested and compared with the supported "IF *<variable name>*" script command. They can be compared for actual contents, or just tested for existence, regardless of the contents.

You will receive an error message under any of these instances and a return code is set which can be tested by your processing, variable errors are not considered fatal:

You attempt to create a variable that already exists

You attempt to REPLACE the contents of a variable that does not yet exist.

If you attempt to DELETE a variable that does not yet exist, you will receive an error, but it is informational only.

Possible Return codes from sub-commands:

DELETE:

VRC=0	OK
VRC=4	Delete Failure
VRC=9	Command failure (user error, see messages)

SET/CREATE:

VRC=0	OK
VRC=2	Created with blanks (no actual variable found)
VRC=4	Variable already existed, replaced with new value
VRC=8	Create failure
VRC=9	Command failure (user error, see messages)

REPLACE:

VRC=0	OK
VRC=2	Variable was replaced with BLANKS, no actual value provided
VRC=6	Variable no longer viable (probable z/OS system error) (deleted)
VRC=7	Variable no longer viable (probable z/OS system error) (still exists)
VRC=8	Variable did not already Exist, not created
VRC=9	Command failure (user error, see messages)

Parameters:

up to 16 byte variable contents (begins with first non-blank)

Example:

```
.....
REPVART <BRI-10-123> 0123456789ABCDEF
REPVARNP <BRI-10-9AB>  ABCDEFGHIJKLMNOPQRS <truncates at "P"
REPTVAR <CICSisUP>  &MM/DD-&HH:MM:SS (results in 11/23-01:21:13)
...
WTO CICS started at &<CICSisUP>
***** this sends a WTO stating "CICS started at 11/23-01:21:13" to the console
...
** if CICS is up is set then start something, $ON$ just checks that it's set to anything
**  $OFF$ checks if it not set to anything, i.e. does not exist.
IF <CICSisUP> =  $ON$
    Go do something
```

```
ELSE
    Go do something else
ENDIF
....
```


DELVAR – Delete any user Variable (Both NON-Persistent & Persistent)

<Variable name>

“DELVARP” is an alias for DELVAR

“DELPVAR” is an alias for DELVAR

“DELTVAR” is an alias for DELVAR

“DELVART” is an alias for DELVAR

“DELVARNP” is an alias for DELVAR

The DELVAR control command the way to delete any user created variable (both Persistent and NON-Persistent). All formats of the DELVAR command (DELVART, DELVARP, etc.) are functionally the same. There is no difference made between the persistence of a variable when it is deleted.

The variable name “<variable name>” can be made up from any ebcdic supported characters, upper and lower case, up to 12 bytes in length not including the “<>” (greater than/less than) characters. Spaces within the name are supported. i.e. <TST_VARIABLE> (12 bytes) is preferable to <TST VARIABLE>, but both are supported. Also note that <Test_Var> and <TEST_VAR> are **completely** different variables because of the upper/lower case character differences.

Example: If you were processing messageID IEF403I for CICS001:

```
IEF403I CICS001 - STARTED - TIME=20.06.17
W00   W01       W03       W05
```

If you were to specify “DELVART <MY_&W01>”

SyzMPF/z will Delete a variable named **“MY_CICS001”**

If found it will be deleted, if not found a Return Code is set.

Variables can also be tested and compared with the supported “IF <variable name>” script command. They can be compared for actual contents, or just tested for existence, regardless of the contents.

You will receive an error message under any of these instances and a return code is set which can be tested by your processing, variable errors are not considered fatal:

You attempt to create a variable that already exists

You attempt to REPLACE the contents of a variable that does not yet exist.

If you attempt to DELETE a variable that does not yet exist, you will receive an error, but it is informational only.

Possible Return codes from sub-commands:

DELETE:

VRC=0	OK
VRC=4	Delete Failure
VRC=9	Command failure (user error, see messages)

SET/CREATE:

VRC=0	OK
VRC=2	Created with blanks (no actual variable found)
VRC=4	Variable already existed, replaced with new value
VRC=8	Create failure
VRC=9	Command failure (user error, see messages)

REPLACE:

VRC=0	OK
VRC=2	Variable was replaced with BLANKS, no actual value provided
VRC=6	Variable no longer viable (probable z/OS system error) (deleted)
VRC=7	Variable no longer viable (probable z/OS system error) (still exists)
VRC=8	Variable did not already Exist, not created
VRC=9	Command failure (user error, see messages)

Parameters:

none

Example:

```
.....  
DELVART <BRI-10-123>  
DELVARNP <BRI-10-9AB>  
RDELVAR <CICSisUP>  
...
```

&LVRC Display the Last Variable Return Code

The &LVRC control command is technically not a command at all, but is the representation of the Last Variable Return Code which was set by the previous Variable Command. If you were to create, replace or delete a new <variable> a return code is set based on the results of that function. You can use the &LVRC variable to display that variable or use it to build other commands. The &LVRC is a single byte return code between 0 and 9. You can perform IF nesting commands based on the "IF LVRC" command, but the &LVRC allows you to display it visually.

Parameters:

NONE

Example: Message is → ABC1234I This is a test of the parse2 command

.....

<TEST_VAR> CREATE THIS is a TeSt

WTO The create of <TEST_VAR> was RC=&LVRC and contains "&<TEST_VAR>"

**** Results in the following WTO:

The create of <TEXT_VAR> was RC=4 and contains "THIS is a TeSt"

*** RC=4 means that the Variable had already existed and we replaced the value

*** had we used REPLACE instead of CREATE, the RC would have been zero.

.....

Possible Return codes from sub-commands which can be checked via the IF LVRC command or displayed via the &LVRC variable:

DELETE:

VRC=0 OK

VRC=4 Delete Failure

VRC=9 Command failure (user error, see messages)

CREATE:

VRC=0 OK

VRC=2 Created with blanks (no actual variable found)

VRC=4 Variable already existed, replaced with new value

VRC=8 Create failure

VRC=9 Command failure (user error, see messages)

REPLACE:

VRC=0	OK
VRC=2	Variable was replaced with BLANKS, no actual value provided
VRC=6	Variable no longer viable (probable z/OS system error) (deleted)
VRC=7	Variable no longer viable (probable z/OS system error) (still exists)
VRC=8	Variable did not already Exist, not created
VRC=9	Command failure (user error, see messages)

&Wnn | &Wn a specific word of message being processed

The &Wnn (or &Wn for single digit word numbers) control command is technically not a command at all, but is the representation of a WORD of the message (from &W00 (zero-zero) that is normally the MessageID being processed, up to &W49 (the 50th word of the message), that was saved in the &Wnn variable by the "PARSE2words" command (Discussed later in this manual). These variables can be used anywhere within any command line of the script, and the "&Wnn" is replaced by the contents of the saved variable "&Wnn". As of version 5.5, this can be a single digit word, so &W05 and &W5 are the same word offset.

Remember, words are counted from ZERO, not ONE. Normally the messageID (message number) is word zero (&W00 or &W0).

Parameters:

NONE

Example: Message is → ABC1234I This is a test of the parse2 command

.....
WTO &W7 &W8 &W04 &W5 &W0
**** Results in the following WTO:
"Parse command test of ABC1234I"

.....

&WORD

The &WORD control command is actually not a command at all, but is the representation of the WORD of the message that was saved in a variable by the "GET WORD" command (Discussed later in this manual). This variable can be used anywhere within any line of the script, and the "&WORD" is replaced by the contents of the saved variable "&WORD".



Note: The &Wnn command is a better choice for this function, it provides the same capability but (when used with the PARSE2WORDS command will set up to 50 words of the message being processed all at once and they can be used without recursive GET commands.

Parameters:

NONE

Example: Message is:

	00	01	02	03	04	05	06
	ABC12345	This	is	a	test	system	UPDATE

.....

*

GET WORD 06 (The word we captured into the variable is the word "UPDATE")

..... lots of other commands

S TEST,M=&WORD (Results in "S TEST,M=UPDATE")

.....

::labelname

labelname = any (up to) 8 Characters

The ::labelname control command is used to allow branching to specific points in the command script. The destination of the “GOTO” branch is identified by the “::labelname” command where “labelname” is any (up to) 8 characters. This creates an anchor so that SyzMPF/z will know that it “possibly” might be branching here in the future. There is no executable code associated with this command, it just sets up a “label” so that the GOTO command has somewhere to go::labelname that is the object of the GOTO.

**** See GOTO for more information.

Parameters:

None

Example:

```
.....
Some misc commands logic
.....
GOTO=CICS2 ----->-----v
::CICS1      <<<<----labelname = CICS1
IF STARTED CICS001
  (EXIT)
ELSE
  Do some other commands
ENDIF
(EXIT)
::CICS2      <<<----labelname = CICS2-----
IF STARTED CICS002
  (EXIT)
ELSE
  Do some other commands
ENDIF
.....
```

AMRF ON/OFF

The AMRF control command is used to cause the SyzMPF/z scripting language to set automatic RETAIN for the MPF message being processed (or turn it off if AMRFOFF is used).

Parameters:

NONE

Example:

```
.....  
* Auto Retain this message  
AMRFON
```

```
.....
```


ASIS

The ASIS control command is used to cause the SyzMPF/z scripting language to bypass the processing of &WORD, &STR (or &STRING), or &REPLID in a script line.

If for some reason, you need to actually use the characters "&WORD", "&STR", "&STRING", or "&REPLYID" in a scripting line, placing the ASIS command at the beginning of the line will cause SyzMPF/z to treat that line exactly as it was written, (i.e. do NOT replace the variable).

Parameters:

NONE

Example:

```
.....  
*  
GET WORD 06      (The word we captured into the variable is the word "UPDATE")  
  
S TEST,M=&WORD    (Results in "S TEST,M=UPDATE")  
  
ASIS S TEST,M=&WORD    (Results in "S TEST,M=&WORD")  
.....
```

AUTOMATE

The AUTOMATE control command is used to cause the SyzMPF/z scripting language to pass the message on to other message automation packages after we are done with it. This can be handy in cases where you are replacing one of the other super expensive message processing packages with this, much better alternative, and wish to keep them both active for a period of time.

Parameters:

NONE

Example:

.....

* pass message on to "the other guy" when we are done

AUTOMATE

.....

AUTOOFF

The AUTOOFF control command is used to cause the SyzMPF/z scripting language to NOT pass the message on to other message automation packages after we are done with it. This is the opposite of the AUTOMATE command.

Parameters:

NONE

Example:

```
.....  
* DON'T pass message on to "the other guy" when we are done  
AUTOOFF  
  
.....
```

DEBUG

Yes | No (no is the default)

The "DEBUG" parameter is used to tell SyzMPF/z whether or not to display DEBUGging information on the console and in the LOG. This command is used to turn off some debugging information such as the creation and deletion of email packages and other messages which may not be desired by the site. This is not the same as "QUIET" although it is similar. **This script command can also be used as a GLOBAL (parmlib).**

Parameters:

Yes | No (no is the default)

Example:

```
.....  
*display email package messages  
DEBUG=YES  
.....
```

ECHO

Script Command Echoing.

SyzMPF/z now supports the ability for users to specify whether or not, or what kinds of Script echoing they wish to have accomplished. Previous to Version 1.2 of the SyzMPF/z product, all commands (System console, and scripting language (IF/ELSE, etc.) were echoed to the system console, or syslog if MPF were used to keep them from the console. We have now added the ability for the user to reduce these messages or completely eliminate them. Issued System Commands will still be issued and logged to syslog and some will show up on the console, but the echoing of those commands and the scripting language can be eliminated or reduced via the new ECHO global command. This GLOBAL command can be entered anywhere within the command script or in the EXEC parms of the job. This GLOBAL ECHO parameter can be specified multiple times within the same script to turn on or off certain types of echoing of commands. The settings are cumulative, so you can add or subtract them in any order or in separate command requests (one per line)

Adding{

- A | All
- I | Ifs
- W|Warnings
- E|Errors
- G|Global
- B|Bypassed
- P|Parse
- C|Commands }

subtracting{

- NONE or OFF**
- NOI |NOIfs**
- NOG|NOGlobal**
- NOW|NOWarnings**
- NOE|NOErrors**
- NOB |NOBypassed**
- NOP |NOParse**
- NOC|NOCommands }**

The ECHO global control command is used to control whether or not and to what extent that the script commands, logic and issued commands are echoed to the console and syslog.

Parameters:

A | All
I | Ifs
W|Wait
G|Global
B |Bypassed
P |Parsed
C|Commands
N | None or O|Off
NOI |NOIfs
NOG|NOGlobal
NOW|NOWait
NOB |NOBypassed
NOP |NOParsed
NOC|NOCCommands

A or All - Causes SyzMPF/z to ECHO all commands and script control logic to console.
This was the default before Version 2.0

NONE or OFF - Is the opposite of ALL, it will eliminate all ECHOing of the commands.

G or Global - Causes SyzMPF/z to ECHO the global scripting commands to the console (currently those are limited to ECHO and EOSM). This is the default setting along with C.

NOGlobal - is the opposite of Global.

I or Ifs - Causes SyzMPF/z to ECHO IF-Related logic to the console and syslog, (IF STARTED, IF SYSID, ELSE, ENDIF etc.)

NOIfs - is the opposite of Ifs.

C or Commands- Causes SyzMPF/z to ECHO the issued system commands to the console and syslog, This is the default setting along with G.

NOCCommands - is the opposite of Commands.

W or Wait - Causes SyzMPF/z to ECHO Wait-Related logic to the console and syslog, (DELAY, PAUSE, WAIT UNTILxxxx, etc.) *** Note: Currently not supported

NOWait - is the opposite of Wait. *** Note: Currently not supported

B or Bypass - Causes SyzMPF/z to Bypassed-Related logic to the console and syslog, (IF clauses that fail, ELSE clauses that fail, etc.)

NOBypass - is the opposite of Bypass.

P or Parsed - Causes SyzMPF/z to send the individual parsed out words in response to the PARSE2WORDS command on the console.

NOParsed - is the opposite of Parsed.

Example:

```
.....  
ECHO=NONE    <— First turn off all echoing  
ECHO=Commands <— Then turn on only command echoing  
.....
```

ELSE

The ELSE control command is used to be able to provide alternative processing logic for the IF commands. This ELSE will allow commands that follow it to be processed if the preceding IF statement were found to be opposite in testing, i.e. FALSE processing of a TRUE request. If there was no preceding IF... command, this command will be cause a script failure.

Parameters:

NONE

Example:

```
.....  
* See if we are running on the TEST LPAR the SYSID would be "TEST"  
IF SYSID PROD  
    Command  
    Command  
ELSE          (*Note: The SYSID was not "PROD" so do this processing)  
    Command  
ENDIF  
.....
```


EMAIL and SMS/Text related Commands

EMAIL, SENDMAIL, To, From, ReplyTo, Cc, Bcc, Subject, Msg, TimeZone, MaxCC, StepCC, Etc.

Version 4 and higher of SyzMPF/z provides the ability to send email or SMS text (collectively referred to as "Mail packages") from any script. There are two MAIN Mail commands, "EMAIL" which begins the process of generating sending the mail package, and "SENDMAIL" which terminates the eMail input process and actually begins the process of building and sending the eMail package to the SyzMail/z component. SyzMAIL/z is responsible for the actual physical movement of the mail package to the SMTP server and out to the recipient email address(es) and/or to SMS text recipients.

While most of the mail package related commands (to, from, subject, replyto, cc, bcc, etc.) can be used anywhere within a SyzMPF/z script, the actual email package MUST have at least one "EMAIL" command to begin the process of gathering all of the data generated by those other email related commands, and is normally closely followed by the "MSG:" command, which builds the body of the email. The MSG: command may be of any length and is free form. It may contain variables (which are resolved at script execution time), and is terminated by the SENDMAIL command. Aside from ending the email MSG body, the SENDMAIL command also starts the actual email creation process, building the actual SMTP related commands that make up the email or SMS text.

Example:

.....(\$HASP395 – JES2 end of JOB) member contents...

```

If MSGTYPE JOB          ← if this is a Batch job
  If MAXCC > 0000        ← If the job is other than MaxCC of zero
    EMAIL  MAXCC  STEPCC  ← Send the Maximum and all step CC's
    To: &NOTIFY          ← Send the email to whoever is on the NOTIFY= of jobcard
    MSG: &TASKNAME (&TASKID) ended with a non-zero return code.
      The step return codes are contained in the email, but you may
      Also want to look into the jobs output .
    SENDMAIL
  ELSE                  ← else the job DID end with zero CC
    EMAIL MAXCC          ← just send the JOB ended CC message and data
    To: &NOTIFY
    Cc: &EXECCLASS       ← send copy to production job log
    SENDMAIL
  ENDIF
ENDIF

```

.....

Email:**MAXCC or STEPCC or STEPCC(stepname) or STEPCC(stepname.procstep)
or MSGONLY**

The email command is used to begin the process of gathering all of the data generated by other email related commands, and begins the process of gathering the required data from those other commands to make them available to send an email or SMS text. The EMAIL command can be used to send a free form email, where the user specifies TO:, From: MSG:, SUBJECT: and any other desired parameters, or it can use some pre-existing parms of its own, i.e. MAXCC, which will build the subject, obtain the tasks execution related information, and send it to wherever the site wants it to go. A second EMAIL related parameter "STEPCC" can be used to generate the detail STEP related information for the entire task (or as much of the task as has already finished execution), and send it within the body of the email. The STEPCC can also limit the steps to specific ones of the task, or specific step and proc-steps of the task by specifying the specific step or step.procstep they wish to have noted.

MSGONLY sends the email message without the JOBNAME or the JOBID for those times when you merely want to send an email based on something that happened but you don't need or want the JOBNAME or JOBID of the task that the email was created from shown in the email.

Parameters:

MAXCC	Generates the Maxcc and task related information Subject is also generated dynamically, and may be overridden, if desired.
STEPCC	Generates detail Step related information, including: Step name, program, step CC, etc. (stepname) or (Stepname.procstep) may also be supplied to STEPCC

Example:

... (IRA200E AUXILIARY STORAGE SHORTAGE)...

IF MAXCC > 4

← if the Maximum CC is greater than 4

To: Operator

← Operations nickname

Cc: &USERID

← and whoever owns this job

Cc: TECHSUPT

← Technical support group nickname

Bcc: Errlog

← send this to the error log address

From: &SYSID

← use this systems SYSID as the nickname

Cc: 1234567890@txt.att.net

← text the on-duty person

EMAIL MAXCC STEPCC

← send the max and step cc's

ATTACH=JESMSG LG

← Attach the JES message log

ELSE	← MaxCC < 4
EMAIL MAXCC STEPCC	← just send the email, no attachments
To: &USERID	← Just send to the owner
ENDIF	
SENDMAIL	← Send to one or the other
...	

EndMessage or EndMsg

No other text should be provided on this command line.

The "ENDMSG:" command is used to temporarily (or fully) end the current message being created within the script. At times it may be advantageous to cease the creation of an email message and to be able to start it later. Normally the SENDMAIL command would end the message, but you can supply ENDMNSG or ENDMESSAGE by itself on a line to cause the message creation process to stop. You can then use a new MSG: command to restart the message and it will be concatenated to the previous message(s).

Parameters:

NONE are allowed

Example:

```
... (IRA200E AUXILIARY STORAGE SHORTAGE)...
To: Operator                ← Operations nickname
Cc: TECHSUPT                ← Technical support group nickname
Bcc: Errlog                 ← send this to the error log address
From: &SYSID                 ← use this systems SYSID as the nickname
Cc: 1234567890@txt.att.net      ← text the on-duty person
MSG: AUX storage Shortage!!
ENDMSG                    ← this temporarily ends the message
IF <PAGEADDED> $OFF$        ← See if we were here already
    SETVAR <PAGEADDED> &MM/DD/YY at &HH:&MM&SS    ← set to current
    PAGEADD First.extra.page.data.set    ← add an additional page dataset
    Subject: &MSGID Storage Shortage (pageDS was added)
    EMAIL
    SENDMAIL
ELSE                          ← Not our first time here
    Subject: &MSGID storage shortage happened again (last page DS added)
    Cc: MANAGER              ← this time make sure the manager knows also
```

```
PAGEADD Second.extra.page.dataset
EMAIL
MSG: A second page dataset was added at &HH:&MM:&SS on &MM/DD/YY
A previous page dataset was added on &<PAGEADDED>. There
are no more page datasets to add.
SENDMAIL
ENDIF
...
```

This example results (the first time it gets called), in sending an email to the OPERATOR nickname (which can be any number of operations personnel), the TechSupt nickname, (also any number of systems programmers), and a SMS/text is sent to the on duty person. The subject of these email/text will be:

[IRA200E Storage Shortage \(pageDS was added\)](#)

And it will contain "AUX storage Shortage!!" in the email/text body.

The next time this message occurs, (we set a variable the first time), the same personnel will get the second email, and this time we have added the manager's nickname (which can be the email and/or SMS text address for them). We have changed the Subject to:

[AUX storage Shortage!!](#)
[IRA200E storage shortage happened again \(last page DS added\)](#)

And we have added a message within the body of the email/text that states:

[AUX storage Shortage!!](#)
[A second page dataset was added at 11:23:58 on 01/02/14](#)
[A previous page dataset was added on 10/20/14 at 01:02:03. There](#)
[are no more page datasets to add.](#)

SENDMAIL

There are no parameters for this command

The sendmail command is used to end the email generation process and begin the process of actually sending the email to the recipients. SENDMAIL will check to make sure that there are sufficient parameters to build the email or sms/text. It will not check that the nicknames (if used) are valid, only that the necessary elements of the email composition, (T), subject, from, etc.) are specified. The actually resolving of the nicknames are performed by SyzMAIL/z once SyzMPF/z is done creating the email package.

Parameters:

None

Example:

```
... (IRA200E AUXILIARY STORAGE SHORTAGE)...
```

To: Operator	← Operations nickname
Cc: TECHSUPT	← Technical support group nickname
Bcc: Errlog	← send this to the error log address
From: &SYSID	← use this systems SYSID as the nickname
Cc: 1234567890@txt.att.net	← text the on-duty person
IF <PAGEADDED> \$OFF\$	← See if we were here already
SETVAR <PAGEADDED> &MM/DD/YY at &HH:&MM&SS	← set to current
PAGEADD First.extra.page.data.set	← add an additional page dataset
Subject: &MSGID Storage Shortage (pageDS was added)	
EMAIL	
SENDMAIL	
ELSE	← Not our first time here
Subject: &MSGID storage shortage happened again (last page DS added)	
Cc: MANAGER	← this time make sure the manager knows also
PAGEADD Second.extra.page.dataset	
EMAIL	
MSG: A second page dataset was added at &HH:&MM:&SS on &MM/DD/YY	
A previous page dataset was added on &<PAGEADDED>. There	
Are no more page datasets to add.	
SENDMAIL	
ENDIF	
...	

This example results (the first time it gets called), in sending an email to the OPERATOR nickname (which can be any number of operations personnel), the TechSupt nickname, (also any number of systems programmers), and a SMS/text is sent to the on duty person. The subject of these email/text will be:

[IRA200E Storage Shortage \(pageDS was added\)](#)

And it will contain no other data in the email/text body.

The next time this message occurs, (we set a variable the first time), the same personnel will get the second email, and this time we have added the manager's nickname (which can be the email and/or SMS text address for them). We have changed the Subject to:

[IRA200E storage shortage happened again \(last page DS added\)](#)

And we have added a message within the body of the email/text that states:

[A second page dataset was added at 11:23:58 on 01/02/14](#)
[A previous page dataset was added on 10/20/14 at 01:02:03. There are no more page datasets to add.](#)

To:**Email Address or Nickname or &Variable (up to 60 bytes)**

The "To:" command is used to provide an email address or nickname (nicknames are described in the SyzMAIL/z manual) to use for "this" outgoing eMail or SMS/text message. This setting is not required if the site wishes to use the "default" setting supplied in the SyzSUBss startup parameter member of the system parmlib concatenation. This can be set to a variable such as "To: ¬ify", which will send the email or SMS text message to the nickname that matches the NOTIFY= parameter of the JOBcard or /* NOTIFY JECL parameter. Any valid variable may be used, but be sure to code a corresponding nickname in the SyzMAIL/z nicknames dataset (please see the SyzMail/z manual for more information on nicknames).

Parameters:

Email Address or Nickname or &Variable

Example:

```
... (IRA200E AUXILIARY STORAGE SHORTAGE)...
To: Operator                ← Operations nickname
Cc: TECHSUPT                 ← Technical support group nickname
Bcc: Errlog                  ← send this to the error log address
From: &SYSID                 ← use this systems SYSID as the nickname
Cc: 1234567890@txt.att.net    ← text the on-duty person
IF <PAGEADDED> $OFF$         ← See if we were here already
    SETVAR <PAGEADDED> &MM/DD/YY at &HH:&MM&SS    ← set to current
    PAGEADD First.extra.page.data.set ← add an additional page dataset
    Subject: &MSGID Storage Shortage (pageDS was added)
    EMAIL
    SENDMAIL
ELSE                          ← Not our first time here
    Subject: &MSGID storage shortage happened again (last page DS added)
    Cc: MANAGER               ← this time make sure the manager knows also
    PAGEADD Second.extra.page.dataset
    EMAIL
    MSG: A second page dataset was added at &HH:&MM:&SS on &MM/DD/YY
        A previous page dataset was added on &<PAGEADDED>. There
        Are no more page datasets to add.
    SENDMAIL
ENDIF
```

...

This example results (the first time it gets called), in sending an email to the OPERATOR nickname (which can be any number of operations personnel), the TechSupt nickname, (also any number of systems programmers), and a SMS/text is sent to the on duty person. The subject of these email/text will be:

[IRA200E Storage Shortage \(pageDS was added\)](#)

And it will contain no other data in the email/text body.

The next time this message occurs, (we set a variable the first time), the same personnel will get the second email, and this time we have added the manager's nickname (which can be the email and/or SMS text address for them). We have changed the Subject to:

[IRA200E storage shortage happened again \(last page DS added\)](#)

And we have added a message within the body of the email/text that states:

[A second page dataset was added at 11:23:58 on 01/02/14](#)
[A previous page dataset was added on 10/20/14 at 01:02:03. There are no more page datasets to add.](#)

From:**Email Address or Nickname or &Variable (up to 60 bytes)**

The "From:" command is used to provide an email address or nickname (nicknames are described in the SyzMAIL/z manual) to use for "this" outgoing eMail or SMS/text message. This setting is not required if the site wishes to use the "default" setting supplied in the SyzSUBss startup parameter member of the system parmlib concatenation. This can be set to a variable such as "From: ¬ify", which will send the email or SMS text message From the nickname that matches the NOTIFY= parameter of the JOBCard or /* NOTIFY JECL parameter. Any valid variable may be used, but be sure to code a corresponding nickname in the SyzMAIL/z nicknames dataset (please see the SyzMail/z manual for more information on nicknames).

Parameters:

Email Address or Nickname or &Variable

Example:

```
... (IRA200E AUXILIARY STORAGE SHORTAGE)...
To: Operator                ← Operations nickname
From: Critical-ERROR@thissite.com ← Override the default
Cc: TECHSUPT                ← Technical support group nickname
Bcc: Errlog                 ← send this to the error log address
From: &SYSID                ← use this systems SYSID as the nickname
Cc: 1234567890@txt.att.net    ← text the on-duty person
IF <PAGEADDED> $OFF$        ← See if we were here already
    SETVAR <PAGEADDED> &MM/DD/YY at &HH:&MM&SS    ← set to current
    PAGEADD First.extra.page.data.set    ← add an additional page dataset
    Subject: &MSGID Storage Shortage (pageDS was added)
    EMAIL
    SENDMAIL
ELSE                        ← Not our first time here
    Subject: &MSGID storage shortage happened again (last page DS added)
    Cc: MANAGER             ← this time make sure the manager knows also
    PAGEADD Second.extra.page.dataset
    EMAIL
    MSG: A second page dataset was added at &HH:&MM:&SS on &MM/DD/YY
        A previous page dataset was added on &<PAGEADDED>. There
        Are no more page datasets to add.
    SENDMAIL
ENDIF
```

...

This example results (the first time it gets called), in sending an email to the OPERATOR nickname (which can be any number of operations personnel), the TechSupt nickname, (also any number of systems programmers), and a SMS/text is sent to the on duty person. The subject of these email/text will be:

IRA200E Storage Shortage (pageDS was added)

And it will contain no other data in the email/text body.

The next time this message occurs, (we set a variable the first time), the same personnel will get the second email, and this time we have added the manager's nickname (which can be the email and/or SMS text address for them). We have changed the Subject to:

IRA200E storage shortage happened again (last page DS added)

And we have added a message within the body of the email/text that states:

A second page dataset was added at 11:23:58 on 01/02/14
A previous page dataset was added on 10/20/14 at 01:02:03. There
are no more page datasets to add.

Cc:**Email Address or Nickname or &Variable (up to 60 bytes)**

The "CC:" command is used to provide an email address or nickname (nicknames are described in the SyzMAIL/z manual) to use for "this" outgoing eMail or SMS/text message. This setting is not required if the site wishes to use the "default" setting supplied in the SyzSUBss startup parameter member of the system parmlib concatenation. This can be set to a variable such as "Cc: ¬ify", which will send the email or SMS text message to the nickname that matches the NOTIFY= parameter of the JOBcard or /* NOTIFY JECL parameter. Any valid variable may be used, but be sure to code a corresponding nickname in the SyzMAIL/z nicknames dataset (please see the SyzMail/z manual for more information on nicknames).

Parameters:

Email Address or Nickname or &Variable

Example:

```
... (IRA200E AUXILIARY STORAGE SHORTAGE)...
To: Operator                      ← Operations nickname
Cc: TECHSUPT                    ← Technical support group nickname
Bcc: Errlog                      ← send this to the error log address
From: &SYSID                    ← use this systems SYSID as the nickname
Cc: 1234567890@txt.att.net      ← text the on-duty person
IF <PAGEADDED> $OFF$            ← See if we were here already
    SETVAR <PAGEADDED> &MM/DD/YY at &HH:&MM&SS      ← set to current
    PAGEADD First.extra.page.data.set      ← add an additional page dataset
    Subject: &MSGID Storage Shortage (pageDS was added)
    EMAIL
    SENDMAIL
ELSE                              ← Not our first time here
    Subject: &MSGID storage shortage happened again (last page DS added)
    Cc: MANAGER                  ← this time make sure the manager knows also
    PAGEADD Second.extra.page.dataset
    EMAIL
    MSG: A second page dataset was added at &HH:&MM:&SS on &MM/DD/YY
        A previous page dataset was added on &<PAGEADDED>. There
        Are no more page datasets to add.
    SENDMAIL
ENDIF
...
```

This example results (the first time it gets called), in sending an email to the OPERATOR nickname (which can be any number of operations personnel), the TechSupt nickname, (also any number of systems programmers), and a SMS/text is sent to the on duty person. The subject of these email/text will be:

[IRA200E Storage Shortage \(pageDS was added\)](#)

And it will contain no other data in the email/text body.

The next time this message occurs, (we set a variable the first time), the same personnel will get the second email, and this time we have added the manager's nickname (which can be the email and/or SMS text address for them). We have changed the Subject to:

[IRA200E storage shortage happened again \(last page DS added\)](#)

And we have added a message within the body of the email/text that states:

[A second page dataset was added at 11:23:58 on 01/02/14](#)
[A previous page dataset was added on 10/20/14 at 01:02:03. There are no more page datasets to add.](#)

Bcc: Blind carbon copy**Email Address or Nickname or &Variable (up to 60 bytes)**

The "Bcc:" command is used to provide an email address or nickname (nicknames are described in the SyzMAIL/z manual) to use for "this" outgoing eMail or SMS/text message. The ID specified will not show up as a recipient on any other recipients email other than the one specified here. This setting is not required if the site wishes to use the "default" setting supplied in the SyzSUBss startup parameter member of the system parmlib concatenation. This can be set to a variable such as "Bcc: ¬ify", which will send the email or SMS text message to the nickname that matches the NOTIFY= parameter of the JOBCard or /* NOTIFY JECL parameter. Any valid variable may be used, but be sure to code a corresponding nickname in the SyzMAIL/z nicknames dataset (please see the SyzMail/z manual for more information on nicknames).

Parameters:

Email Address or Nickname or &Variable

Example:

```
... (IRA200E AUXILIARY STORAGE SHORTAGE)...
To: Operator                   ← Operations nickname
Cc: TECHSUPT                  ← Technical support group nickname
Bcc: Errlog                   ← send this to the error log address
From: &SYSID                   ← use this systems SYSID as the nickname
Cc: 1234567890@txt.att.net   ← text the on-duty person
IF <PAGEADDED> $OFF$          ← See if we were here already
   SETVAR <PAGEADDED> &MM/DD/YY at &HH:&MM&SS   ← set to current
   PAGEADD First.extra.page.data.set   ← add an additional page dataset
   Subject: &MSGID Storage Shortage (pageDS was added)
   EMAIL
   SENDMAIL
ELSE                           ← Not our first time here
   Subject: &MSGID storage shortage happened again (last page DS added)
   Cc: MANAGER               ← this time make sure the manager knows also
   PAGEADD Second.extra.page.dataset
   EMAIL
   MSG: A second page dataset was added at &HH:&MM:&SS on &MM/DD/YY
       A previous page dataset was added on &<PAGEADDED>. There
       Are no more page datasets to add.
   SENDMAIL
ENDIF
```

...

This example results (the first time it gets called), in sending an email to the OPERATOR nickname (which can be any number of operations personnel), the TechSupt nickname, (also any number of systems programmers), and a SMS/text is sent to the on duty person. The subject of these email/text will be:

[IRA200E Storage Shortage \(pageDS was added\)](#)

And it will contain no other data in the email/text body.

The next time this message occurs, (we set a variable the first time), the same personnel will get the second email, and this time we have added the manager's nickname (which can be the email and/or SMS text address for them). We have changed the Subject to:

[IRA200E storage shortage happened again \(last page DS added\)](#)

And we have added a message within the body of the email/text that states:

[A second page dataset was added at 11:23:58 on 01/02/14](#)
[A previous page dataset was added on 10/20/14 at 01:02:03. There are no more page datasets to add.](#)


```
MSG: A second page dataset was added at &HH:&MM:&SS on &MM/DD/YY
      A previous page dataset was added on &<PAGEADDED>. There
      Are no more page datasets to add.
SENDMAIL
ENDIF
...
```

This example results (the first time it gets called), in sending an email to the OPERATOR nickname (which can be any number of operations personnel), the TechSupt nickname, (also any number of systems programmers), and a SMS/text is sent to the on duty person. The subject of these email/text will be:

[IRA200E Storage Shortage \(pageDS was added\)](#)

And it will contain no other data in the email/text body.

The next time this message occurs, (we set a variable the first time), the same personnel will get the second email, and this time we have added the manager's nickname (which can be the email and/or SMS text address for them). We have changed the Subject to:

[IRA200E storage shortage happened again \(last page DS added\)](#)

And we have added a message within the body of the email/text that states:

[A second page dataset was added at 11:23:58 on 01/02/14](#)
[A previous page dataset was added on 10/20/14 at 01:02:03. There](#)
[are no more page datasets to add.](#)

SUBJect: or Subj:

Any text (including &variables, up to 75 bytes)

The "Subj:" command is used to provide subject text to use for "this" outgoing eMail or SMS/text message. This setting is not required but if left out will result in a subject of "no subject supplied" being used. This can be set to a variable such as "Subj: &MSGID from &TASKNAME (&TASKID)" (results in (for IEF123I message) a subject in the email of: IEF123I from CICS123 (STC0169)). Any valid variable may be used, but be sure to code a corresponding nickname in the SyzMAIL/z nicknames dataset (please see the SyzMail/z manual for more information on nicknames).

Parameters:

Any text or valid &variable

Example:

```
... (IRA200E AUXILIARY STORAGE SHORTAGE)...
To: Operator                ← Operations nickname
Cc: TECHSUPT                ← Technical support group nickname
Bcc: Errlog                 ← send this to the error log address
From: &SYSID                 ← use this systems SYSID as the nickname
Cc: 1234567890@txt.att.net  ← text the on-duty person
IF <PAGEADDED> $OFF$        ← See if we were here already
    SETVAR <PAGEADDED> &MM/DD/YY at &HH:&MM&SS    ← set to current
    PAGEADD First.extra.page.data.set ← add an additional page dataset
    Subject: &MSGID Storage Shortage (pageDS was added)
    EMAIL
    SENDMAIL
ELSE                        ← Not our first time here
    Subject: &MSGID storage shortage happened again (last page DS added)
    Cc: MANAGER            ← this time make sure the manager knows also
    PAGEADD Second.extra.page.dataset
    EMAIL
    MSG: A second page dataset was added at &HH:&MM:&SS on &MM/DD/YY
        A previous page dataset was added on &<PAGEADDED>. There
        Are no more page datasets to add.
    SENDMAIL
ENDIF
...
```

This example results (the first time it gets called), in sending an email to the OPERATOR nickname (which can be any number of operations personnel), the TechSupt nickname, (also any number of systems programmers), and a SMS/text is sent to the on duty person. The subject of these email/text will be:

[IRA200E Storage Shortage \(pageDS was added\)](#)

And it will contain no other data in the email/text body.

The next time this message occurs, (we set a variable the first time), the same personnel will get the second email, and this time we have added the manager's nickname (which can be the email and/or SMS text address for them). We have changed the Subject to:

[IRA200E storage shortage happened again \(last page DS added\)](#)

And we have added a message within the body of the email/text that states:

[A second page dataset was added at 11:23:58 on 01/02/14](#)
[A previous page dataset was added on 10/20/14 at 01:02:03. There are no more page datasets to add.](#)

Message: or Msg:

Any text (including &variables) (no maximum length except individual line length, number of lines is unlimited))

The "MSG:" command is used to supply the body of an email. There is no limit to the number of lines after the MSG: command, and the message may start on the same line as the MSG: command. Blank lines and comments may be supplied, but depending on the location in the script, they may be ignored. Variables may be used, and if they are valid (i.e. they exist) then they will be resolved at execution time. The MSG: command is the last command before the SENDMAIL command, and is ended by the inclusion of the SENDMAIL command. Future versions of SyzMPF/z will allow IF/THEN/ELSE statements to be used within the MSG block, but at this time it is not supported. If no MSG: command is provided, then no text will be included in the email or SMS/Text message. Any valid variable may be used, but be sure to code a corresponding nickname in the SyzMAIL/z nicknames dataset (please see the SyzMail/z manual for more information on nicknames). Messages may be temporarily (or fully) ended with the endmsg or endmessage command (described later)

Parameters:

Any text or valid &variable

Example:

```
... (IRA200E AUXILIARY STORAGE SHORTAGE)...
```

To: Operator	← Operations nickname
Cc: TECHSUPT	← Technical support group nickname
Bcc: Errlog	← send this to the error log address
From: &SYSID	← use this systems SYSID as the nickname
Cc: <u>1234567890@txt.att.net</u>	← text the on-duty person
IF <PAGEADDED> \$OFF\$	← See if we were here already
SETVAR <PAGEADDED> &MM/DD/YY at &HH:&MM&SS	← set to current
PAGEADD First.extra.page.data.set	← add an additional page dataset
Subject: &MSGID Storage Shortage (pageDS was added)	
EMAIL	
SENDMAIL	
ELSE	← Not our first time here
Subject: &MSGID storage shortage happened again (last page DS added)	
Cc: MANAGER	← this time make sure the manager knows also
PAGEADD Second.extra.page.dataset	
EMAIL	
MSG: A second page dataset was added at &HH:&MM:&SS on &MM/DD/YY	

**A previous page dataset was added on &<PAGEADDED>. There
Are no more page datasets to add.**

```
SENDMAIL  
ENDIF  
...
```

This example results (the first time it gets called), in sending an email to the OPERATOR nickname (which can be any number of operations personnel), the TechSupt nickname, (also any number of systems programmers), and a SMS/text is sent to the on duty person. The subject of these email/text will be:

[IRA200E Storage Shortage \(pageDS was added\)](#)

And it will contain no other data in the email/text body.

The next time this message occurs, (we set a variable the first time), the same personnel will get the second email, and this time we have added the manager's nickname (which can be the email and/or SMS text address for them). We have changed the Subject to:

[IRA200E storage shortage happened again \(last page DS added\)](#)

And we have added a message within the body of the email/text that states:

[A second page dataset was added at 11:23:58 on 01/02/14](#)
[A previous page dataset was added on 10/20/14 at 01:02:03. There
are no more page datasets to add.](#)

MAXMSG =**nnnn**

The "MAXMSG" control command is used to tell SyzMPF/z how many lines to pre-allocate for the current email message being produced. It will remain in effect until the next MAXMSG or the end of the script. This may have been set by the startup parameter in parmlib, but can be overridden for an individual script. In particular if the EMAIL MAXCC or STEPCC is being used, and the task has many steps, it could easily run out of space to list the detail for each step. The default is 50 lines.

Parameters:**9999 (50 lines is the default)****Example:**

```
.... (IRA200E AUXILIARY STORAGE SHORTAGE)...
To: Operator                   ← Operations nickname
Cc: TECHSUPT                 ← Technical support group nickname
Bcc: Errlog                  ← send this to the error log address
From: &SYSID                 ← use this systems SYSID as the nickname
MAXMSG=150                   ← up to 150 lines instead of default(50)
Cc: 1234567890@txt.att.net     ← text the on-duty person
IF <PAGEADDED> $OFF$         ← See if we were here already
    SETVAR <PAGEADDED> &MM/DD/YY at &HH:&MM&SS     ← set to current
    PAGEADD First.extra.page.data.set     ← add an additional page dataset
    Subject: &MSGID Storage Shortage (pageDS was added)
    EMAIL
    SENDMAIL
ELSE                         ← Not our first time here
    Subject: &MSGID storage shortage happened again (last page DS added)
    Cc: MANAGER              ← this time make sure the manager knows also
    PAGEADD Second.extra.page.dataset
    EMAIL
    MSG: A second page dataset was added at &HH:&MM:&SS on &MM/DD/YY
        A previous page dataset was added on &<PAGEADDED>. There
        Are no more page datasets to add.
    SENDMAIL
ENDIF
```

ATTACH =

JESJCL | JESMSG LG | JESYSMSG | JESJCLIN | JESALL | DSN | JESDSID=nnn

The "ATTACH" control command is used to tell SyzMPF/z which JES related datasets are to be attached to the email package. You can specify any or all of the JES datasets (or "JESALL" to send them all) or DSN=datasetname or DSN=PDSdatasetname(member) to send either a sequential dataset or a member of a PDS. The individual JES datasets are separated by identification and the total lines in each dataset is presented at the end of the dataset.

Parameters:

JESJCL - Attaches the JCL images as output from the converter

JESMSG LG - Attaches the JES message log (WTO's issued by the task)

JESYSMSG - Attaches the system messages (JCL resolution messages and condition codes)

JESJCLIN - Attaches the JCL as submitted or started. (JCL before JES processing)

JESALL - Attaches all of the above datasets (JESJCL, JESMSG LG, JESYSMSG and JESJCLIN)

DSN - Attaches any sequential dataset or member of a PDS.

JESDSID=nnn Send any specific JES2 DSSID from the job

DSID=001, 002, 003 and 004 are "special"

DSID=001 is the JESJCLIN

DSID=002 is the JESMSG LG

DSID=003 is the JESJCL

DSID=004 is the JESYSMSG

You should use the names above instead of the DSID for these datasets

Example:

... (IRA200E AUXILIARY STORAGE SHORTAGE)...

IF MAXCC > 4

← if the Maximum CC is greater than 4

To: Operator

← Operations nickname

Cc: &USERID

← and whoever owns this job

Cc: TECHSUPT

← Technical support group nickname

Bcc: Errlog	← send this to the error log address
From: &SYSID	← use this systems SYSID as the nickname
Cc: <u>1234567890@txt.att.net</u>	← text the on-duty person
EMAIL MAXCC STEPCC	← send the max and step cc's
ATTACH=JESMSG LG	← Attach the JES message log
ELSE	← MaxCC < 4
EMAIL MAXCC STEPCC	← just send the email, no attachments
To: &USERID	← Just send to the owner
ENDIF	
SENDMAIL	← Send to one or the other
...	

ENDIF

The ENDIF control command is used to end a list of conditional commands set by a preeceding IF.... command. If there was no preceding IF.... command, this command will be ignored.

Parameters:

NONE

This command will end the if clause of the preceding IF or nested IF command.

Example:

```
.....  
* See if we are running on production  
IF SYSID PROD  
    Command  
    Command  
ELSE  
    Command  
ENDIF  
.....
```


(EXIT)

The EXIT and control command is used to stop (or abort) the execution of the script in progress.

Parameters:

None

Example:

```
.....  
* IF CICS001 is running then don't execute these commands at all and set RC=4.  
IF STARTED CICS001  
(EXIT)  
ELSE  
    Do some other commands  
ENDIF  
.....
```

GET REPLY

GETREPLY

The GET REPLY control command is used to cause the command script to set a variable (&REPLY) to contain the numeric REPLY ID of the WTOR MPF message being processed. This variable can be inserted at any location on any line FOLLOWING the GET command. (See the &REPLY command later in this manual). If the message being processed is NOT a WTOR, an error results and no REPLYID variable is set.

There are no IF/THEN/ELSE requirements for this script command, it sets the variable &REPLYID so that it can be used any number of times later in the same script. Only one &REPLYID variable can be set at a time. Subsequent GETREPLY commands will replace the variable with the new contents.

Parameters:

None

Example:

```
.....  
*  
GET REPLY                (The REPLY ID of the WTOR we are processing is saved)  
GET WORD 06              (The word we captured into the variable is the word  
"UPDATE")  
..... lots of other commands  
  
R &REPLY,&WORD  
  
.....
```

GET STRing

@xx(y) | @ xx(y) | x(y) (xx=offset yy=Length)

GETSTRING

The GET STRINGD control command is used to cause the command script to set a variable (&STRING or &STR) to contain the text of the STRING at a specific offset and length within the MPF message being processed. This variable can be inserted at any location on any line FOLLOWING the GET command. (See the &STR or &STRING command later in this manual). The STRING beginning offset must be specified as a 1 or 2-digit numeric relative to 0 (zero) and should be preceded by the special character at-sign "@". The STRING length is specified as a 1 or 2-digit numeric (up to 64 bytes). The first BYTE or any message (BYTE zero (00) is ALWAYS the first character of the message ID itself. For instance, the 7 words of the following message are identified below the message (Case sensitive):

```
SYZSP21E - Multiple Blank Lines were skipped.
0         1         2         3         4         5
012345678901234567890123456789012345678901234567890
```

```
Offset 10 is "-"
Offset 20 is "e"
```

```
STRING @ 22(5) is "Blank"
STRING @28(15) is "Lines were skip"
```

There are no IF/THEN/ELSE requirements for this script command, it sets the variable &STR (or &STRING) so that it can be used any number of times later in the same script. Only one &STR(or &STRING) variable can be set at a time. Subsequent GETSTRING commands will replace the variable with the new contents.

Parameters:

- @ xx(yy)
- @ - Optional – Suggested to denote that we are starting "at" the next spot
- xx - required - the offset (beginning with zero) of the string to be saved as a variable .
- yy - required – the length of the string to be used (**begins with '1', NOT '0'**)
- (-) - suggested use would be to use the beginning and ending parentheses',
beginning is required ending is option but suggested.

Example (IEE391A is a Multi-Line WTO, but is treated as a single line by GET):

```
.....
* 0           1           2           3           6           5
* 012345678901234567890123456789012345678901234567890123
* IEE391A SMF ENTER DUMP FOR DATA SET ON VOLSER SYSRES,
      DSN=SYS1.MANB
*           5           6
*           4567890123456
```

GET STRING 54(13)

* start the SMFDUMP task

IF SYSID PROD

 S SMFTEST,&STR (Results in S SMFTEST,DSN=SYS1.MANB

ELSE

 S SMFPROD,&STR (Results in S SMFPROD,DSN=SYS1.MANB

ENDIF

.....

GET WORD

nn STR xx(yy)
or
nn @ xx(yy)

GETWORD

The GET WORD control command is used to cause the command script to set a variable (&WORD) to contain the text of the WORD at a specific WORD offset within the MPF message being processed. This variable can be inserted at any location on any line FOLLOWING the GET command. (See the &WORD command later in this manual). "WORD"s are delimited by at least one blank space. The WORD number must be specified as a 1 or 2-digit numeric relative to 0 (zero). The first WORD or any message (WORD zero (00) is ALWAYS the message ID itself. For instance, the 7 words of the following message are identified below the message (Case sensitive). The word can be a substring of that same word by using the STR sub-command along with the start position (xx) and the length of the substring-ed word. The default length is the remaining part of that word, for instance, if word 7 of a message were SYSTEMX and the STRing was set to 3(2), that word when used later would be set to "TE":

```
SYZSP21E - Multiple Blank Lines were skipped.  
  (00)   (01)   (02)   (03) (04)  (05)  (06)  
Word 00 = SYZSP21E  
Word 01 = - (hyphen)  
Word 02 = Multiple  
Word 03 = Blank  
Word 04 = Lines  
Word 05 = were  
Word 06 = Skipped
```

There are no IF/THEN/ELSE requirements for this script command, it sets the variable &WORD so that it can be used any number of times later in the same script. Only one &WORD variable can be set at a time. Subsequent GETWORD commands will replace the variable with the new contents.

Parameters:

nn
nn - required - the WORD offset (beginning with zero) of the word to be saved as a variable.
STR - Optional – sets the WORD to a substring of that word
Xx - Optional – The offset for the beginning of the substring (beginning at '0')

(yy)- required (if offset is used) - the STRING length of the text to be saved as a variable (**beginning at '1' NOT '0'**). The default is the remaining length of the word

OR

nn - required - the WORD offset (beginning with zero) of the word to be saved as a variable.

@ - optional – tells SYZMPF that we will be limiting the &WORD variable to some substring of the original word

Xx - Optional – The offset for the beginning of the new word based on the old word starting position of '0'

(yy)- required (if offset is used) - the STRING length of the text to be saved as a variable (**beginning at '1' NOT '0'**). The default is the remaining length of the word

Example (IEE391A is a Multi-Line WTO, but is treated as a single line by GET):

.....

```
* WORD(0) (1) (2) (3) (4) (5) (6) (7) (8) (9) (10)
* IEE391A SMF ENTER DUMP FOR DATA SET ON VOLSER SYSRES, DSN=SYS1.MANB
GET WORD 10
* start the SMFDUMP task
IF SYSID PROD
    S SMFPROD,&WORD (Results in S SMFPROD,DSN=SYS1.MANB)
ELSE
    S SMFPROD,&WORD (Results in S SMFPROD,DSN=SYS1.MANB)
ENDIF
```

****Substring could also be used

```
* 0 1 2 3 6 5
* 012345678901234567890123456789012345678901234567890123
* IEE391A SMF ENTER DUMP FOR DATA SET ON VOLSER SYSRES,
* WORD(0) (1) (2) (3) (4) (5) (6) (7) (8) (9)
    DSN=SYS1.MANB
    (10)
* 5 6
* 456789012345

*WORD 10 = DSN=SYS1.MANB
* 1
* 0123456789012
```

GET WORD 10 STR 12(1) ← word 10 starting at the 12th byte of that word for 1 byte

IF SYSID=PROD

S SMFPROD,DSN=SYS1.MAN&WORD (results in S SMFPROD,DSN=SYS1.MANB)

ELSE

S SMFTTEST,DSN=SYS1.MAN&WORD (results in S SMFTTEST,DSN=SYS1.MANB)

ENDIF

.....

GOTO

labelname

The GOTO control command is used to allow branching to specific points in the command script. The destination to branch to is the “::labelname” and is required, unless the ++RESET option is used. SyzMPF/z has automatic LOOP detection code, and the ++RESET option will turn that code off for the processing after the ::labelname branch has started. The “GOTO ++RESET” command should occur (if used) somewhere after the ::labelname that is the object of the GOTO.

**** See ::labelname for more information.

Parameters:

Labelname - required. The name of the label to branch to within the script.

Example:

```
.....
Some misc commands logic
.....
GOTO=CICS2 ----->-----v
|
::CICS1      <<<<----labelname = CICS1  |
IF STARTED CICS001                        |
  (EXIT)                                |
ELSE                                       |
  Do some other commands                 |
ENDIF                                    |
(EXIT)                                   |
::CICS2      <<<-----labelname = CICS2-----|
IF STARTED CICS002
  (EXIT)
ELSE
  Do some other commands
ENDIF
.....
```


HIGHLIGHT

The HIGHLIGHT control command is used to cause the SyzMPF/z scripting language to highlight the MPF message being processed on the operator console. This command can also be issued in the "AUTO(HIGHLIGHT)" part of the actual MPFLSTxx member, thereby negating the need to have a corresponding SyzMPF/z MPF member to be processed

Parameters:

NONE

Example:

In MPFLST MEMBER:

```
.....  
DFHSM013*,SUP(NO),USEREXIT(SYZMPFZ),AUTO(HIGHLIGHT)  
.....
```

OR in a SyzMPFz MPF script for the same DFHSM013* series of messages

```
.....  
* highlight the messages  
HIGHLIGHT
```

```
.....
```

“IF” Based Command general Information:

Almost all IF-Based commands now support command stacking and multi-level IF/AND command stacking. The concept of command stacking is nothing new, SyzMPF/z has supported IF-Nesting since its inception which allowed you to perform multiple IF's before actually performing the actions you want to take. Command Stacking allows you to perform the same functions as IF/THEN/ELSE-Nesting, without all of the extra work. You can stack any number of command options (as long as they fit on a single line). This restriction will be removed in future releases.

If you are comparing Variables (thing that start with an “&” as in “&W1” (word 1 of the message), and that variable is blank or “empty”, then your IF statement can be wildly different from what you intend. For instance:

IF &W3 ne FRED works great when the third word of the message is some actual word or character. So for message “IBM123456 THIS task is special”, &W3 is the word “is” (&W0 is the messageID IBM123456).

But if the message were “IBM123456 Nothing happened”, then there is no third word of the message, so &W3 is null, so now the IF statement takes on a whole different context:

Now it's “IF ne FRED”. When SyzMPF/z takes this command apart to service it, we have the first field to compare is now the characters “ne”, then SyzMPF/z searches for a comparator (=,^=,>,<,eq,ne,gt,lt), and when it doesn't see one it “assumes” that you meant “equal”, and uses the FRED as the second field to compare.

So, now you are comparing the characters ‘ne’ to the characters ‘FRED’, which is still false, but not really what you had intended. When used with “ne” or “^=” you can see that it would be very simple to end up with unintended consequences. So when using variables, be certain that you take into account for the possibility of a null entry.

Simply stating “IF XX&W3 ne XXFRED would resolve this because if it was null you would be asking: IF XX ne XXFRED, and if the &W3 was indeed “FRED” then it would be “IF XXFRED ne XXFRED, which would really be truly the “false” you were looking for



Previous to version 5 of Syzmpf/z, If you wanted to send an email only if the MaxCC of the task was greater than 2 but less than 8, you had to do that in two steps (see example 1):

Example 1 IF nesting method:

```
IF Maxcc > 2
  IF MaxCC < 8
    EMAIL MAXCC
    TO: somebody@SyzygyInc.com
    SENDMAIL
  ENDIF
ENDIF
```

As of Version 5.2, you can perform this same work with one line (see Example 2):

```
IF Maxcc > 2 + < 8      ← maxcc greater than 2 and-less-than 8
  EAMIL MAXCC
  TO somebody@syzygyinc.com
  SENDMAIL
ENDIF
```

Example 2 actually shows another important feature of the stacked commands, in that normally between parameters you would use the “|” (or) or the “+” (and) symbol alone, but if you wanted to change the “default” that you set in the first parameter to a new comparator, (i.e. in this case from Greater-than to Less-then), you can do that by combining or concatenating the new comparator to the |/+ separator, thereby changing the logic from one type of comparator to another. Another command stacking separator that is supported from version 5.2 and above is the “||” (orIF) and the “++” (andIF) separator. These are used when you want to stack a set of commands, but those commands are not the same “IF-COMMAND”. For instance, you may want to do something if the MaxCC is greater than some value, but ONLY IF we are executing on a specific LPAR, again, you could already do this with multiple IF/THEN/ELSE-nesting, but it takes a lot of space and is sometimes cumbersome to use, (see Example 3):

Example 3:

```
IF LPAR = PROD
  IF MAXCC ne ABEND
    IF MAXCC > 2
      IF MAXCC < 8
```

```
EMAIL MAXCC
TO somebody@SyzygyInc.com
SENDMAIL
ENDIF
ENDIF
ENDIF
ENDIF
```

But with the new support in version 5.2 you can do this all at one time (see Example 4):

Example 4:

```
IF LPAR = PROD ++ IF MAXCC ne ABEND |> 2 |< 8
EMAIL MAXCC
TO: Somebody@SyzygyInc.com
SENDMAIL
ENDIF
```

There are various comparators that can be used are:

=, EQ, eq, Eq, eQ	← means “EQUAL”
^=, NE, ne, Ne, nE	← means “NOT EQUAL”
>, GT, gt, Gt, gT	← means “GREATER THAN”
<, LT, lt, Lt, lT	← means “LESS THAN”
/, GE, ge, Ge, gE	← means “GREATER THAN OR EQUAL TO”
\, LE, le, Le, lE	← means “LESS THAN OR EQUAL TO”

One important note is that when you are switching the comparators in command stacking, you MUST concatenate them to the AND or OR separator (identified below). If you fail to do that, the result will be that the comparator last used will be in place still, and you will try to compare the field to the new comparator. For instance:

Bad command:

```
IF LPAR = PROD1 | PROD2 | prOD3 | test2 > TEST5
```

Does NOT compare the LPAR to PROD1, or PROD2, or PROD 3 or TEST4 or GREATER THAN TEST5. Instead it does the following:

Compares the LPAR to PROD1, or PROD2, or PROD3, or TEST1 or “>” (the greater than symbol, the rest of the line is ignored).

Obviously this is NOT what you desired to do, and had you concatenated the OR symbol and the GREATER-THAN symbol to “|>”, it would have worked as desired. It should have been coded as:

Correct command:

```
IFLPAR = PROD1 | PROD2 | prOD3 | test2 ≥ TEST5
```

Along with the new comparators, there are also various separators that can be used. At this time, they are limited to 4 separators, but that will change over time:

	← means “OR”
+	← means “AND”
	← means “ORif”
++	← means “ANDif”

The “|” (or) and the “+” (and) separators can be used in conjunction with the comparators to change the meaning of the compare, but do not effect the object of the compare. For instance, if you code:

```
IF LPAR = PROD | TEST | SANDBOX | BRIAN
```

The LPAR is tested for any of the 4 options being true, but the base object “LPAR” has not changed.

If you code the “||” (ORif) or the “++” (andIF) separators, they MUST be followed by a COMPLETE IF statement, even if that IF statement is the same as the first one. Normally the “||” (orIF) and “++” (andIF) separators are used when you need to change the things you are comparing for, i.e. LPAR and TASKNAME, or even LAPR and LPAR if you wish, but if you use the “||” (orif) or “++” (andif) commands, they must be followed by FULL IF commands, not just the compared fields:

Example 5:

```
IFLPAR = TEST ++ IFTASKNAME = PROD*  
  Do something  
ELSE  
  Do something else  
ENDIF
```

Example 5 above, checks that we are on the “TEST” LPAR and any task that starts with the characters “PROD”. This might seem odd, but many sites use this sequence for their \$HASP373

processing (job started) on their TEST LPARs, so that if or some reason a “PRODUCTION” job were to start on the TEST LPAR, that it can be cancelled and queued to the PROD LPAR instead, or just plain cancelled because they want to keep bad things from happening to production jobs. Similarly, they use the same type of processing on their production LPAR to keep test jobs from executing on them. It’s very effective.

The important thing to remember is that if you are going to use two dissimilar types of commands on the same line, you have to separate them with the “||” (orif) or “++” (Andif) separators so that SyzMPF/z will know that it is supposed to process a “NEW” type of command and not just reprocess the old command with a new operator object.

IF “ANYTHING” Compare any variable, text, string, etc.

=|EQ|^=|NE|>|GT|<|LT|/|GE|\\LE “ANYTHING ELSE”

“OR”, “|”, “AND”, and “+” are supported in this command
“ORIF”, “||”, “ANDIF”, “++” supported by this command

The IF “ANYTHING” control command is used to cause the command script to test for the existence and/or contents of the “ANYTHING” variable specified. You can also specify another &variable or anything else within the contents of the “value” to be tested for. This command begins an “IF NEST” of commands that will be done only IF the condition being tested for is a TRUE statement. If the condition is found to not be TRUE (as requested), the condition is set to FALSE, and all subsequent commands until the next ELSE or the matching ENDIF statement are bypassed. The ELSE statement will allow commands that should be executed on a FALSE condition to be executed. Indentation of IF levels is not required, but makes the script more readable to a human.

Parameters:

“ANYTHING” The compare value
“ANYHING ELSE” the value to compare to

Example:

```
.....
*Make sure we are executing after 10PM on February 23rd
* &MN=Month, &DD=DAY, &HH=hour,&MM=minute
...
IF &MN/&DD = 02/23
*** do some stuff if we are Feb 23rd
ELSE
*** do something else it's not Feb 23rd
ENDIF
...
```

IF <variable> (Generics Supported)

\$ON\$

\$OFF\$

=|EQ Value (up to 16 characters including other variables and blanks)

^|=|NE Value (up to 16 characters including other variables and blanks)

>|GT Value (up to 16 characters including other variables and blanks)

<|LT Value (up to 16 characters including other variables and blanks)

/|GE Value (up to 16 characters including other variables and blanks)

\|LE Value (up to 16 characters including other variables and blanks)

IF <Variable>

The IF <variable> control command is used to cause the command script to test for the existence and/or contents of the <variable> specified. If you supply a parameter of \$ON\$, then you are asking to check to see if the variable called <variable> exists. Supplying a sub-command of \$OFF\$ is the same as the ELSE conditional of the \$ON\$ sub-command, in other words that the <variable> is not set or not used at all. You can also specify another variable within the contents of the "value" to be tested for. For instance, assuming you previously had set the <My_TEST> variable to the value "MON" and you were to test that value via a "IF <MY_TEST> EQ &DOW (day of week), then you would get a true condition on Mondays, but a false on every other day. This command begins an "IF NEST" of commands that will be done only IF the condition being tested for is a TRUE statement. If the condition is found to not be TRUE (either EQ, NE, \$ON\$ or \$OFF\$ as requested), the condition is set to FALSE, and all subsequent commands until the next ELSE or the matching ENDIF statement are bypassed. The ELSE statement will allow commands that should be executed on a FALSE condition to be executed. Indentation of IF levels is not required, but makes the script more readable to a human.



<variable> Persistent variables can be created at any time in any script and checked from any other script, they do not have to be set in the script that they are being used in, which makes it a great way to pass long term parameters between scripts that may have nothing to do with each other except that they happen to be able to use information within those variables. NON-Persistent variables are just as useful, with the added benefit that when the task that they were attached to goes away, the variable goes away as well.

Parameters:

\$ON\$ The <Variable> is set to something (even blanks are acceptable).

\$OFF\$	The <variable> is not used or not set to anything.
= EQ VALUE	The <variable> is set to EXACTLY the VALUE
^= NE VALUE	The <variable> is not set to EXACTLY the VALUE
> GT Value	The <variable> is Greater than the VALUE
< LT Value	The <variable> is Less Than the VALUE
/ GE Value	The <variable> is Greater than or EQUAL to the VALUE
\ LE Value	The <variable> is Less than or EQUAL to the VALUE

Example:

```
.....
* Create variable <MY_LPAR> is set to the LPAR name "SYST"
SETVAR <MY_LPAR> &LPAR ←-this is on the SYST lpar
...
IF <MY_LPAR> EQ SYSA
*** do some stuff if this is SYSTEM A
ENDIF
...
IF <MY_LPAR> EQ SYST
*** do stuff if this is the SYST LPAR
ENDIF
.....
```

IF ACTive | IF STARTed (Generics Supported)

**=|eq|ne|^=
TASKNAME**

“ORIF”, “||”, “ANDIF”, “++” supported by this command

IFACTIVE

Alias of IF STARTed, See IF STARTED section for more information.

IF AFTER **(Replaced by IF TIME, phasing command out)****HH:MM****IF AFTER**

The IF AFTER control command is used to cause the command script to test for the current, time of day and see if the current time is AFTER the time specified on the control command. You are required to identify the Time of Day in HH (Hours) and MM (minute) format that you wish to test for. You may not specify more than one specific time. This command begins an "IF NEST" of commands that will be done only IF the condition being tested for is a TRUE statement. If the current Time is found to not be AFTER the time you have selected, the condition is set to FALSE, and all subsequent commands until the next ELSE or the matching ENDIF statement are bypassed. The ELSE statement will allow commands that should be executed on a FALSE condition to be executed, (i.e. the current time is **not** AFTER the requested time of day you have specified in the control command). Indentation of IF levels is not required, but makes the script more readable to a human.

Parameters:**HH:MM**

Time of day in 24 hour format 00:00 through 23:59.

Example:

```
.....
* Make sure it's before 8am or after 5pm
IF BEFORE 08:00
    $P PRT15
    $PI
    $TI1-5,ABCD
    $SI1-5
ELSE
    IF AFTER 17:00
        $PI
        $TI1-5,CDEF
        $SI1-5
    ENDIF
ENDIF
.....
```

IF BEFORE (Replaced by IF TIME, being phased out)**HH:MM****IFBEFORE**

The IF BEFORE control command is used to cause the command script to test for the current, time of day and see if the current time is BEFORE the time specified on the control command. You are required to identify the Time of Day in HH (Hours) and MM (minute) format that you wish to test for. You may not specify more than one specific time. This command begins an "IF NEST" of commands that will be done only IF the condition being tested for is a TRUE statement. If the current Time is found to not be before the time you have selected, the condition is set to FALSE, and all subsequent commands until the next ELSE or the matching ENDIF statement are bypassed. The ELSE statement will allow commands that should be executed on a FALSE condition to be executed, (i.e. the current time is **not** BEFORE the requested time of day you have specified in the control command). Indentation of IF levels is not required, but makes the script more readable to a human.

Parameters:**HH:MM**

Time of day in 24 hour format 00:00 through 23:59.

Example:

```
.....
* Make sure it's before 8am or after 5pm
IF BEFORE 08:00
    $P PRT15
    $PI
    $TI1-5,ABCD
    $SI1-5
ELSE
    IF AFTER 17:00
        $PI
        $TI1-5,CDEF
        $SI1-5
    ENDIF
ENDIF
.....
```

IF CLASS (Generics Supported)

X **=|EQ|^=|NE|>|GT|<|LT|/|GE|\|LE**
(where 'x' is any valid class 0-9 or a-z)
--- NULLS or (NULL represents no CLASS provided (STC)
"OR", "|", "AND", and "+" are supported in this command
"ORIF", "||", "ANDIF", "++" supported by this command

IFCLASS

The IF CLASS control command is used to cause the command script to test for the JOB or execution class of the issuer of the message being processed. You are required to identify the single byte CLASS that you wish to test for. You may not specify more than one class at a time. This command begins an "IF NEST" of commands that will be done only IF the condition being tested for is a TRUE statement. If the Class you have elected to test for is found to not be the actual execution class, the condition is set to FALSE, and all subsequent commands until the next ELSE or the matching ENDIF statement are bypassed. The ELSE statement will allow commands that should be executed on a FALSE condition to be executed, (i.e. the requested class is NOT the class you are testing for). Indentation of IF levels is not required, but makes the script more readable to a human.

Parameters:

X **(where 'x' is any valid class 0-9 or A-Z)**

Example:

```
.....
* See if we are executing in production class P or Class X
IF CLASS = P | X
    Wto Production JOB &TASKNAME is running
    .... (other commands)
ELSE
    WTO &taskname was not a production job because it ran in class=&EXECCLASS
ENDIF
.....
```

IF COMMAND

=|eq|ne|^=

Yes|No

“OR”, “|”, “AND”, and “+” are supported in this command

“ORIF”, “||”, “ANDIF”, “++” supported by this command

IFCOMMAND

The IF COMMAND control command is used to cause the command script to test for the issuer of the message that we are processing currently. If the message was issued from a JOB or problem program as a normal WTO, then the setting will be “NO”. If the message we are processing were entered by an operator command or as an internal operator command (i.e. issued by SyzCMD/z or SyzMPF/z, TSO CONSOLE, SDSF, or some other utility as an operator command, then this will be set to “YES”.

This option can be tested and operator commands can be “built from scratch” because we will know at the time we are processing the script that we are processing a command and not a message.

This command begins an “IF NEST” of commands that will be done only IF the condition being tested for is a TRUE statement. If the message is found to not be issued from a console, the “Yes” condition is set to FALSE, and all subsequent commands until the next ELSE or the matching ENDIF statement are bypassed. The ELSE statement will allow commands that should be executed on a FALSE condition to be executed, (i.e. the current message was not issued from a console or internally as a command and is therefore set to “No”). Indentation of IF levels is not required, but makes the script more readable to a human.

Parameters:

Yes The message was issued as a command

No The message was issued as a normal WTO or WTOR

Example:

```
****MPFLSTnn = “@*,SUP(NO),USEREXIT(SYZMPF30)”
```

```
.....
```

```
* Make our own @STOP CICS command (this scripts name is “@” )
```

```
*make sure this is a command and not just some other @message
```

IF COMMAND EQ Y

```
  If word00 @STOP
    IF WORD01 CICS
      WTO Stopping CICS001
      F CICS001,CEMT P shut
      S syzcmdz,m=stopc001 (make sure it comes down right)
      WTO Stopping CICS002
      F CICS002,CEMT P shut
      S syzcmdz,m=stopc002 (make sure it comes down right)
      WTO Stopping CICS003
      F CICS003,CEMT P shut
      S syzcmdz,m=stopc003 (make sure it comes down right)

    ENDIF
    IF WORD01 TSO
      WTO STOPPING TSO
      S syzcmdz,m=stoptso
    ENDIF
  ENDIF
  IF WORD00 @START
    IF WORD01 CICS
      WTO Starting CICS001
      S CICS001
      S syzcmdz,m=strtc001 (wait for CICS1 to start and bring the rest up)
      WTO Starting CICS001
      S CICS002
      S syzcmdz,m=strtc001 (wait for CICS2 to start and bring the rest up)
      WTO Starting CICS003
      S CICS003
      S syzcmdz,m=strtc003 (wait for CICS3 to start and bring the rest up)
    ENDIF
  ENDIF
ENDIF
.....
```

IF CURRENTTIME | IF TIME (Generics Supported)

=|EQ|^|=|NE|>|GT|<|LT|/|GE|\\LE the “current” time HH:MM
“OR”, “|”, “AND”, and “+” are supported in this command
“ORIF”, “||”, “ANDIF”, “++” supported by this command

IFCURRENTTIME

Alias of IF TIME command, see IFTIME section for more information.

IF DOW (Generics Supported)

=|eq|ne|^=

MONday|TUEsday|WEDnsday|THUrsmday|FRIday|SATurday|SUNDAY

-FIRST | -SECOND | -THIRD | -FOURTH | -FIFTH

“OR”, “|”, “AND”, and “+” are supported in this command

“ORIF”, “||”, “ANDIF”, “++” supported by this command

IFDOW

The IF DOW control command is used to cause the command script to test for the current, specific Day Of the. You are required to identify the Day of the Week that you wish to test for. You may not specify more than one day at a time. This command begins an “IF NEST” of commands that will be done only IF the condition being tested for is a TRUE statement. If the Day you have elected to test for is found to not be the current day of the week, the condition is set to FALSE, and all subsequent commands until the next ELSE or the matching ENDIF statement are bypassed. The ELSE statement will allow commands that should be executed on a FALSE condition to be executed, (i.e. the requested day of the week is NOT the one you are testing for). Indentation of IF levels is not required, but makes the script more readable to a human.

Additionally you can check “which” day of the week for the month it is, i.e. the second Saturday would be “IFDOW = SATURDAY -SECOND” would ONLY match on the second Saturday of the month. Or you can sepecify that you want it to be any day EXCEPT the second Saturday with “IFDOW NE SATURDAY -SECOND”

Parameters:

MONday|TUEsday|WEDnsday|THUrsmday|FRIday|SATurday|SUNday

One of the Days Of the Week are required. You may abbreviate the day to 3 characters or more.

-FIRST | -SECOND | -THIRD | -FOURTH | -FIFTH

One of the above can be used at a time, and must appear AFTER the Day of the week and must begin with a dash ‘-’.

Example:

.....

* Make sure it's Saturday or Sunday

IF DOW SATURDAY

 \$P PRT15

 \$PI

 \$TI1-5,ABCD

 \$SI1-5

ELSE

 IF DOW SUNDAY

```
$PI
$T11-5,CDEF
$S11-5
ENDIF
ENDIF
IF DOW SATURDAY or SUNDAY      (alternate method)
ENDIF
*
* Only the first Sunday of the month
IF DOW = SUNDAY -FIRST
    • Do something
ELSE
    • DO nothing or something else
ENDIF
.....
```

IF INACTIVE | IF IFSTOPped (Generics Supported)

**=|eq|ne|^=
TASKNAME**

“ORIF”, “||”, “ANDIF”, “++” supported by this command

IFINACTIVE

Alias of IF STOPped, See IF STOPped section for more information.

IF LPAR (Generics Supported)

=|EQ|^=|NE|>|GT|<|LT|/|GE|\\LE LPARNAME
NULLS or (NULL represents no LPAR known
“OR”, “|”, “AND”, and “+” are supported in this command
“ORIF”, “||”, “ANDIF”, “++” supported by this command

IFLPAR

The IF LPAR control command is used to cause the command script to test for the LPAR NAME (or generic LPAR name) that the script processor is executing under. You are required to identify the LPAR NAME that you wish to test for. You may not specify more than one LPAR NAME at a time.

This command begins an “IF NEST” of commands that will be done only IF the condition being tested for is a TRUE statement. If the LPAR NAME you have elected to test for is found to not be the current LPAR NAME that we are executing under, the condition is set to FALSE, and all subsequent commands until the next ELSE or the matching ENDIF statement are bypassed. The ELSE statement will allow commands that should be executed on a FALSE condition to be executed, (i.e. the requested LPAR NAME is NOT the one you are testing for). Indentation of IF levels is not required, but makes the script more readable to a human.

Parameters:

LPARNAME

The LPARName may contain the following Generic Characters:

'%' means any single character is to be considered a match
'*' means from that point on is considered a match

The one to 8 character LPAR name (as specified on the HMC) is required.

Example:

```
.....
* Make sure it's the PRODUCTION system
IF LPAR PROD
  $P PRT15
  $PI
  $TI1-5,ABCD
  $SI1-5
ELSE
* not PRODUCTION, so might be TEST
  IF LPAR=TEST
```

```
$PI  
$TI1-5,CDEF  
$SI1-5  
ENDIF  
ENDIF
```

*Do something ONLY if we are not executing on the PROD or TEST LPARs

If LPAR ne PROD or TEST

*not prod or test so we are something else

ENDIF

.....

IF LVRC (Generics Supported)

=|EQ|^=|NE|>|GT|<|LT|/|GE|\\LE

Returncode (Single digit between 0 and 9)

“OR”, “|”, “AND”, and “+” are supported in this command

“ORIF”, “||”, “ANDIF”, “++” supported by this command

IFLVRC

The IF LVRC control command provides that ability to check the return code from the previous Variable Command DElete, CREate or REPlace function. If you were to create, replace or delete a new <variable> a return code is set based on the results of that function. You can use the IF LVRC variable to check that return code. The LVRC is a single byte return code between 0 and 9. You can perform IF nesting commands based on the “IF LVRC” command. You may not specify more than one day at a time. This command begins an “IF NEST” of commands that will be done only IF the condition being tested for is a TRUE statement. If the Last Return Code from a variable command you have elected to test for is found to not be the single digit code you are testing for, the condition is set to FALSE, and all subsequent commands until the next ELSE or the matching ENDIF statement are bypassed. The ELSE statement will allow commands that should be executed on a FALSE condition to be executed, (i.e. the requested return code is NOT the one you are testing for). Indentation of IF levels is not required, but makes the script more readable to a human.

Parameters:

= EQ	Equal to (between 0 and 9)
^= NE	Not Equal to (between 0 and 9)
> GT	Greater than (between 0 and 8)
< LT	Less Than (between 1 and 9)

Example: Message is → ABC1234I This is a test of the parse2 command

```
.....
<TEST_VAR> CREATE THIS is a TeSt
WTO The create of <TEST_VAR> was RC=&LVRC and contains "&<TEST_VAR>"
**** Results in the following WTO:
The create of <TEXT_VAR> was RC=4 and contains "THIS is a TeSt"
*** RC=4 means that the Variable had already existed and we replaced the value
*** had we used REPLACE instead of CREATE, the RC would have been zero.
IF LVRC < 5
*** do some stuff if we created or replaced okay
```

```
ELSE
  WTO We failed to create <TEST_VAR>
  WTO the return code was &LVRC and we needed it to be less than 4
  (EXIT)
ENDIF
```

.....

Possible Return codes from sub-commands which can be checked via the IF LVRC command or displayed via the &LVRC variable:

DELETE:

VRC=0	OK
VRC=4	Delete Failure
VRC=9	Command failure (user error, see messages)

CREATE:

VRC=0	OK
VRC=2	Created with blanks (no actual variable found)
VRC=4	Variable already existed, replaced with new value
VRC=8	Create failure
VRC=9	Command failure (user error, see messages)

REPLACE:

VRC=0	OK
VRC=2	Variable was replaced with BLANKS, no actual value provided
VRC=6	Variable no longer viable (probable z/OS system error) (deleted)
VRC=7	Variable no longer viable (probable z/OS system error) (still exists)
VRC=8	Variable did not already Exist, not created
VRC=9	Command failure (user error, see messages)

IF MaxCC (Generics Supported)

=, EQ, NE, >, GT, <, LT one of the following:
nnnn (1 to 4 character Maximum Condition code to check for)
Sxxx (System Abend)
Unnnn (User Abend)
ABEND (any System or User Abend)
Zero (Condition code of Zero)

IFMAXCC

The IF MAXCC control command is used to cause the command script to test for the Task's Maximum Condition code (so far) of the issuer of the message being processed. The "so far" is because the job may not yet have ended, unless we are processing the \$HASP395 or one of the JOB ENDED messages for the task (IEF404I=normal end, IEF452I=JCL error, IEF450I=ABEND, and others). You are required to identify a numeric or (in the case of an abend), alphanumeric code to check against. You may not specify more than one code at a time. This command begins an "IF NEST" of commands that will be done only IF the condition being tested for is a TRUE statement. If the MaxCC code you have elected to test for is found to not be the actual execution MaxCC (so far), the condition is set to FALSE, and all subsequent commands until the next ELSE or the matching ENDIF statement are bypassed. The ELSE statement will allow commands that should be executed on a FALSE condition to be executed, (i.e. the requested class is NOT the class you are testing for). Indentation of IF levels is not required, but makes the script more readable to a human.

Parameters:

'=' or 'NE' or 'EQ' or '<' or 'LT' or '>' or 'GT'
Nnnn or xxxx (numeric or alphanumeric code)

Example:

* Do some code checking?

IF Maxcc = 0

(EXIT) ← nothing special to do

ELSE

IF MAXCC > S000 ← some type of ABEND S001-SEFF

..... (other commands)

ENDIF

IF Maxcc GT U0000 ← some type of user abend

..... (other commands)

ENDIF

IF MaxCC = ABEND ← any System or User ABEND

..... (other commands)


```
ENDIF
IF maxcc > 11      ← maxcc is 12 or higher
  ..... (other commands)
ENDIF
EMAIL MAXCC STEPCC ← this sends the dynamic maxCC and step cc's
SENDMAIL           ← this releases the email to SyzMAIL
ENDIF
.....
```

IF MAXRESULT (Generics Supported)

=|eq|ne|^=

RESULT (ENDED, ABENDED, FAILED, FLUSHED, ACTIVE)

“OR”, “|”, “AND”, and “+” are supported in this command

“ORIF”, “||”, “ANDIF”, “++” supported by this command

IFMAXRESELT

The IF MAXRESULT (so far if the task has not ended yet) control command is used to cause the command script to test for the tasks result (so far) of the issuer of the message being processed. You are required to identify the result that you wish to test for. You may not specify more than one result at a time. This command begins an “IF NEST” of commands that will be done only IF the condition being tested for is a TRUE statement. If the result you have elected to test for is found to not be the actual result, the condition is set to FALSE, and all subsequent commands until the next ELSE or the matching ENDIF statement are bypassed. The ELSE statement will allow commands that should be executed on a FALSE condition to be executed, (i.e. the requested class is NOT the class you are testing for). Indentation of IF levels is not required, but makes the script more readable to a human.

Parameters:

RESULT (ended, abended, failed, flushed or ACTIVE)

Example:

```
.....
* See if The job ended or abended
IF MAXRESULT = ABENDED
  (do some commands)
ELSE
  IF MAXRESULT = ENDED
    (do some commands)
  ELSE
    WTO The job's result (so far) is &RESULT
  ENDIF
ENDIF
.....
```

IF MAXPROGRAM (Generics Supported)

=|eq|ne|^=

Program Name (1 to 8 characters)

“OR”, “|”, “AND”, and “+” are supported in this command

“ORIF”, “||”, “ANDIF”, “++” supported by this command

IFMAXPROGRAM

The IF MAXPROGRAM control command is used to cause the command script to test for the tasks step program name that had the highest condition code of the issuer of the message being processed. You are required to identify the step program name that you wish to test for. You may not specify more than one step program name at a time. This command begins an "IF NEST" of commands that will be done only IF the condition being tested for is a TRUE statement. If the STEP program name you have elected to test for is found to not be the actual Step program name with the highest condition code, the condition is set to FALSE, and all subsequent commands until the next ELSE or the matching ENDIF statement are bypassed. The ELSE statement will allow commands that should be executed on a FALSE condition to be executed, (i.e. the requested class is NOT the class you are testing for). Indentation of IF levels is not required, but makes the script more readable to a human.

Parameters:

STEP program name

Example:

```
.....
* See if The highest step was executing program ABCDEFG
IF MAXCC > 0000
  IF MAXPROG = ABCDEFG
    (do some commands)
  ELSE
    WTO the highest CC program was &MAXPROGRAM
  ENDIF
ENDIF
.....
```

IF MAXPSTEP

=|eq|ne|^=

PROCSTEPname

“OR”, “|”, “AND”, and “+” are supported in this command

“ORIF”, “||”, “ANDIF”, “++” supported by this command

IFMAXPSTEP

The IF MAXPSTEP control command is used to cause the command script to test for the tasks Proc step that had the highest condition code of the issuer of the message being processed (so far, if the job has not yet ended). You are required to identify the Procstepname that you wish to test for. You may not specify more than one Procstepname at a time. This command begins an “IF NEST” of commands that will be done only IF the condition being tested for is a TRUE statement. If the ProcSTEPname you have elected to test for is found to not be the actual ProcStepname with the highest condition code, the condition is set to FALSE, and all subsequent commands until the next ELSE or the matching ENDIF statement are bypassed. The ELSE statement will allow commands that should be executed on a FALSE condition to be executed, (i.e. the requested class is NOT the class you are testing for). Indentation of IF levels is not required, but makes the script more readable to a human.

Parameters:

PROCSTEPname

Example:

```
.....
* See if The highest Procstep was GO
IF MAXCC > 0000
  IF MAXPSTEP = GO
    (do some commands)
  ENDIF
ENDIF
.....
```

IF MAXSTEP

STEPname

“OR”, “|”, “AND”, and “+” are supported in this command

“ORIF”, “||”, “ANDIF”, “++” supported by this command

IFMAXSTEP

The IF MAXSTEP control command is used to cause the command script to test for the tasks step that had the highest condition code of the issuer of the message being processed (so far, if the job has not yet ended). You are required to identify the stepname that you wish to test for. You may not specify more than one stepname at a time. This command begins an “IF NEST” of commands that will be done only IF the condition being tested for is a TRUE statement. If the STEPname you have elected to test for is found to not be the actual Stepname with the highest condition code, the condition is set to FALSE, and all subsequent commands until the next ELSE or the matching ENDIF statement are bypassed. The ELSE statement will allow commands that should be executed on a FALSE condition to be executed, (i.e. the requested class is NOT the class you are testing for). Indentation of IF levels is not required, but makes the script more readable to a human.

Parameters:

STEPname

Example:

```
.....
* See if The highest step was FREDSTEP
IF MAXCC > 0000
  IF MAXSTEP = FREDSTEP
    (do some commands)
  ENDIF
ENDIF
.....
```

IF MSGCLASS (Generics Supported)

=|EQ|^=|NE|>|GT|<|LT|/|GE|\\LE
X (where 'x' is any valid class 0-9 or a-z)
“OR”, “|”, “AND”, and “+” are supported in this command
“ORIF”, “||”, “ANDIF”, “++” supported by this command

IFMSGCLASS

The IF MSGCLASS control command is used to cause the command script to test for the tasks MSGCLASS setting of the issuer of the message being processed. You are required to identify the single byte MSGCLASS that you wish to test for. You may not specify more than one msgclass at a time. This command begins an “IF NEST” of commands that will be done only IF the condition being tested for is a TRUE statement. If the MSGCLASS you have elected to test for is found to not be the actual msgclass, the condition is set to FALSE, and all subsequent commands until the next ELSE or the matching ENDIF statement are bypassed. The ELSE statement will allow commands that should be executed on a FALSE condition to be executed, (i.e. the requested class is NOT the class you are testing for). Indentation of IF levels is not required, but makes the script more readable to a human.

Parameters:

X (where 'x' is any valid class 0-9 or A-Z)

Example:

```
.....
* See if we are executing with production msgclass P
IF MSGCLASS = P
  Wto Production JOB &TASKNAME is running
  .... (other commands)
ELSE
  WTO &taskname was not a production because it ran with MSGCLASS=&MSGCLASS
ENDIF
.....
```

IF MSGID (Generics Supported)

= MESSAGE ID (up to 25 characters) (generics supported)

The '=' is not required in this request and can be eliminated if desired
"OR", "|", "AND", and "+" are supported in this command
"ORIF", "||", "ANDIF", "++" supported by this command

IFMSGID

The IF MSGID control command is used to cause the command script to test for the specific (or generic) MESSAGE ID of the console message that caused this script to be processed. Normally (for messages up to 8 characters) the script member is only executed for that specific message, but for messages that have more than 8 characters, or those scripts that are executed via the (AUTO) parameter, you might sometimes need to know "exactly" which message is being processed.

This command begins an "IF NEST" of commands that will be done only IF the condition being tested for is a TRUE statement. If the MESSAGE ID you have elected to test for is found to not be the current MESSAGE ID, the condition is set to FALSE, and all subsequent commands until the next ELSE or the matching ENDIF statement are bypassed. The ELSE statement will allow commands that should be executed on a FALSE condition to be executed, (i.e. the requested MESSAGE ID is NOT the one you are testing for). Indentation of IF levels is not required, but makes the script more readable to a human.

Parameters:

MESSAGE ID (up to 25 characters)

The MessageID may contain the following Generic Characters:

'%' means any single character is to be considered a match
'*' means from that point on is considered a match

The one to 25 character MESSAGE ID (as specified in the message that started this script) is required.

Example:

.....For message DFHSI1517

IFMSGID DFHSI1517

* if CICS123 start DB2 Connect A

IFMSGTASK CICS123

```
S DB2CONA
ENDIF
* if CICS456 start DB2 Connect B
IFMSGTASK CICS456
S DB2CONB
ENDIF
ENDIF
.....
```


IF MSGTASK | TASKNAME (Generics Supported)

=|eq|ne|^=

Task Name (up to 8 characters)

The '=' is not required in this request and can be eliminated if desired

"OR", "|", "AND", and "+" are supported in this command

"ORIF", "||", "ANDIF", "++" supported by this command

IFMSGTASK

The IF MSGTASK (or TASKNAME) control command is used to cause the command script to test for the specific (or generic) JOB/STC or TSU task name that issued the message that caused this script to be processed. This command begins an "IF NEST" of commands that will be done only IF the condition being tested for is a TRUE statement. If the TASK NAME you have elected to test for is found to not be the TASK NAME that issued the message that this script is processing, the condition is set to FALSE, and all subsequent commands until the next ELSE or the matching ENDIF statement are bypassed. The ELSE statement will allow commands that should be executed on a FALSE condition to be executed, (i.e. the requested TASK NAME is NOT the one you are testing for). Indentation of IF levels is not required, but makes the script more readable to a human.

Parameters:

TASK NAME (up to 8 characters)

The taskname may contain the following Generic Characters:

'%' means any single character is to be considered a match

'*' means from that point on is considered a match

The one to 8 character TASK NAME is required.

Example:

.....For message DFHSI1517

IF MSGID DFHSI1517

* if CICS123 start DB2 Connect A

IF **MSGTASK** CICS123

S DB2CONA

```
ENDIF
* if CICS456 start DB2 Connect B
  IF TASKNAME = CICS456
    S DB2CONB
  ENDIF
ENDIF
.....
```

IF MSGTYPE

=|eq|ne|^=
Tso, Stc, or Job

IFMSGTYPE

The IF MSGTYPE control command is used to cause the command script to test for the type of task that issued the message. You can test the type of task that issued the command (TSO USER, Started Task or Batch JOB), and act accordingly.

This command begins an "IF NEST" of commands that will be done only IF the condition being tested for is a TRUE statement. If the TASK that issued the message is found to not be the proper type (T for TSO, S for STC, or J for JOB) as tested, the condition is set to FALSE, and all subsequent commands until the next ELSE or the matching ENDIF statement are bypassed. The ELSE statement will allow commands that should be executed on a FALSE condition to be executed, (i.e. the requested TASK is NOT the type requested). Indentation of IF levels is not required, but makes the script more readable to a human.

Parameters:

Tso	Only the first character is required
Stc	
Job	

Example:

```
.....
* Test SYZCM18E mem-leak message to be sure it's from a STC, if TSO user, ignore it
IF MSGTYPE = Tso  ← (or Tso, or TSO or TS, or just T)
* do nothing
    ELSE
* could be are a problem so page Tech Support
S PAGE,USER=TECHSUPP,SEV=1,ERROR="SyzCM18E - Possible memory leak"
ENDIF
.....
```

IF NOTIFY (Generics Supported)

=|EQ|^=|NE|>|GT|<|LT|/|GE|\\LE

Name on the NOTIFY=name parameter

--- NULLS or (NULL represents no Notify ID provided on JOBCARD

“OR”, “|”, “AND”, and “+” are supported in this command

“ORIF”, “||”, “ANDIF”, “++” supported by this command

IFNOTIFY

The IF NOTIFY control command is used to cause the command script to test for the Task's NOTIFY= setting (jobcard or /*JECL card) of the issuer of the message being processed. You are required to identify a "name" that you wish to test for. You may not specify more than one NAME at a time. This command begins an "IF NEST" of commands that will be done only IF the condition being tested for is a TRUE statement. If the notify=NAME you have elected to test for is found to not be the actual execution Notify=name, the condition is set to FALSE, and all subsequent commands until the next ELSE or the matching ENDIF statement are bypassed. The ELSE statement will allow commands that should be executed on a FALSE condition to be executed, (i.e. the requested class is NOT the class you are testing for). Indentation of IF levels is not required, but makes the script more readable to a human.

Parameters:

Name

Example:

```
.....
* See if we are a Payroll group job
IF NOTIFY = NULLS    ←    We could use NULLS or (NULL
  (EXIT)              ← nothing we can check for, so leave
ENDIF
IF Notify = payroll
  Wto Payroll JOB &TASKNAME is running
  .... (other commands)
ELSE
  WTO &taskname was not a payroll job because it has NOTIFY=&notify
ENDIF
.....
```

IF OFFLINE

A=xxxx | V=volser

IFOFFLINE

The IF OFFLINE control command is used to cause the command script to test for a specific UCB address to be OFFLINE or see if a specific Volume Serial is NOT mounted. You are required to identify either the hexadecimal UCB address (A=0123) or the Volume Serial (V=TSO001), but you may not specify both at the same time. This command begins an "IF NEST" of commands that will be done only IF the condition being tested for is a TRUE statement. If the UCB address or VOLSER is found to be online or mounted, the condition is set to FALSE, and all subsequent commands until the next ELSE or the matching ENDIF statement are bypassed. The ELSE statement will allow commands that should be executed on a FALSE condition to be executed, (i.e. the requested UCB or VOLSER **is** online or **IS** mounted). Indentation of IF levels is not required, but makes the script more readable to a human.

Parameters:

A=xxxx

V=volser

One of either A= or V= is required. You may not specify both on the same control command.

A=xxxx - specifies the 1 to 4 hexadecimal address of the UCB in question

v=volser - specifies the volume serial number of the DISK or TAPE volume in question.

Example:

```
.....
* see if MVSRES is mounted, if not mount it and submit backups
IF OFFLINE V=MVSRES
  V A123,ONLINE
  F A,MVSBKUP
ELSE
* If already online, start the backup
  F A,MVSBKUP
ENDIF
.....
```

IF ONLINE

A=xxxx | V=volser

IFONLINE

The IF ONLINE control command is used to cause the command script to test for a specific UCB address to be online or see if a specific Volume Serial is mounted. You are required to identify either the hexadecimal UCB address (A=0123) or the Volume Serial (V=TSO001), but you may not specify both at the same time. This command begins an "IF NEST" of commands that will be done only IF the condition being tested for is a TRUE statement. If the UCB address or VOLSER is found to NOT online or NOT mounted, the condition is set to FALSE, and all subsequent commands until the next ELSE or the matching ENDIF statement are bypassed. The ELSE statement will allow commands that should be executed on a FALSE condition to be executed, (i.e. the requested UCB or VOLSER is NOT online or mounted. Indentation of IF levels is not required, but makes the script more readable to a human.

Parameters:

A=xxxx

V=volser

One of either A= or V= is required. You may not specify both on the same control command.

A=xxxx - specifies the 1 to 4 hexadecimal address of the UCB in question

v=volser - specifies the volume serial number of the DISK or TAPE volume in question.

Example:

```
.....
* see if MVSRES is mounted, if not mount it and submit backups
IF ONLINE V=MVSRES
* If so, start the backup
  F A,MVSBKUP
ELSE
  V A123,ONLINE
  F A,MVSBKUP
ENDIF
.....
```

IF ORIGIN (Generics Supported)

=|EQ|^=|NE|>|GT|<|LT|/|GE|\\LE
SYSTEMID

“OR”, “|”, “AND”, and “+” are supported in this command

“ORIF”, “||”, “ANDIF”, “++” supported by this command

IFORIGIN

The IF ORIGIN control command is used to cause the command script to test for the SYSTEM ID that the actual MPF message originated from. On multi-image systems (i.e. sysplexes) the MPF message may not have originated on the system that is processing the MPF message. This allows the site to check the system name of the system that the MPF message actually was issued upon. You are required to identify the SYSTEMID that you wish to test for. You may not specify more than one SYSTEMID at a time. This command begins an “IF NEST” of commands that will be done only IF the condition being tested for is a TRUE statement. If the SYSTEMID you have elected to test for is found to not be the correct SYSTEM NAME which the MPF message originated on, the condition is set to FALSE, and all subsequent commands until the next ELSE or the matching ENDIF statement are bypassed. The ELSE statement will allow commands that should be executed on a FALSE condition to be executed, (i.e. the requested SYSTEM NAME is NOT the one you are testing for). Indentation of IF levels is not required, but makes the script more readable to a human.

Parameters:

SYSTEMID

The one to 8 character SYSTEM name (as specified in parmlib) is required.

Example:

```
.....  
* Make sure it's the PRODUCTION system  
IF ORIGIN A7PROD  
  $P PRT15  
  $PI  
  $TI1-5,ABCD  
  $SI1-5  
ELSE  
* not PRODUCTION, so must be TEST  
  $PI  
  $TI1-5,CDEF  
  $SI1-5  
ENDIF  
.....
```

IF OSMOD (Generics Supported)

=|EQ|^=|NE|>|GT|<|LT|/|GE|\\LE
MODLEVEL

“OR”, “|”, “AND”, and “+” are supported in this command

“ORIF”, “||”, “ANDIF”, “++” supported by this command

IFOSMOD

The IF OSMOD control command is used to cause the command script to test for the Operating System Modification Level that the script processor is executing under. Z/OS is displayed in the format of VV.RR.MM (i.e. z/OS 01.12.00), where VV=Version number (i.e. 01), RR=Release (i.e. 12), MM=Modification level (i.e. 00). You are required to identify the MODLEVEL that you wish to test for. You may not specify more than one MODLEVEL at a time. This command begins an “IF NEST” of commands that will be done only IF the condition being tested for is a TRUE statement. If the MODLEVEL you have elected to test for is found to not be the current Operating System Modification Level, the condition is set to FALSE, and all subsequent commands until the next ELSE or the matching ENDIF statement are bypassed. The ELSE statement will allow commands that should be executed on a FALSE condition to be executed, (i.e. the requested OS Modification Level is NOT the one you are testing for). Indentation of IF levels is not required, but makes the script more readable to a human.

Parameters:

MODLEVEL

The one to 2 character Operating System Modification Level is required.

Example:

```
.....
* Are we executing under z/OS 1.12.0 or .1 for IMS start?
IF OSVER=01      Check Version 1
  IF OSREL=08      Check Release 8
    S CICSnew
  IF OSMOD=00      Check Mod 00
    S IMS00
  IF OSMOD=01      CHECK Mod 01
    S IMS01
  ENDIF
ENDIF
ENDIF
ENDIF
.....
```


IF OSREL (Generics Supported)

**=|EQ|^=|NE|>|GT|<|LT|/|GE|\\LE
RELEASE**

“OR”, “|”, “AND”, and “+” are supported in this command

“ORIF”, “||”, “ANDIF”, “++” supported by this command

IFOSREL

The IF OSREL control command is used to cause the command script to test for the Operating System Release that the script processor is executing under. Z/OS is displayed in the format of VV.RR.MM (i.e. z/OS 01.12.00), where VV=Version number (i.e. 01), RR=Release (i.e. 12), MM=Modification level (i.e. 00). You are required to identify the Release that you wish to test for. You may not specify more than one Release at a time. This command begins an “IF NEST” of commands that will be done only IF the condition being tested for is a TRUE statement. If the RELEASE you have elected to test for is found to not be the current Operating System Release number, the condition is set to FALSE, and all subsequent commands until the next ELSE or the matching ENDIF statement are bypassed. The ELSE statement will allow commands that should be executed on a FALSE condition to be executed, (i.e. the requested OS Release is NOT the one you are testing for). Indentation of IF levels is not required, but makes the script more readable to a human.

Parameters:

RELEASE

The one to 2 character Operating System Release Number is required.

Example:

```
.....
* Are we executing under z/OS 1.12.0 or .1 for IMS start?
IF OSVER=01      Check Version 1
  IF OSREL=08      Check Release 8
    S CICSnew
  IF OSMOD=00      Check Mod 00
    S IMS00
  IF OSMOD=01      CHECK Mod 01
    S IMS01
  ENDIF
ENDIF
ENDIF
ENDIF
.....
```

IF OSVER (Generics Supported)

**=|EQ|^=|NE|>|GT|<|LT|/|GE|\\LE
VERSION**

“OR”, “|”, “AND”, and “+” are supported in this command

“ORIF”, “||”, “ANDIF”, “++” supported by this command

IFOSVER

The IF OSVER control command is used to cause the command script to test for the Operating System Version that the script processor is executing under. Z/OS is displayed in the format of VV.RR.MM (i.e. z/OS 01.12.00), where VV=Version number (i.e. 01), RR=Release (i.e. 12), MM=Modification level (i.e. 00). You are required to identify the VERSION that you wish to test for. You may not specify more than one VERSION at a time. This command begins an “IF NEST” of commands that will be done only IF the condition being tested for is a TRUE statement. If the VERSION you have elected to test for is found to not be the current Operating System Version number, the condition is set to FALSE, and all subsequent commands until the next ELSE or the matching ENDIF statement are bypassed. The ELSE statement will allow commands that should be executed on a FALSE condition to be executed, (i.e. the requested OS Version is NOT the one you are testing for). Indentation of IF levels is not required, but makes the script more readable to a human.

Parameters:

VERSION

The one to 2 character Operating System Version Number is required.

Example:

```
.....  
* Are we executing under z/OS 1.12.0 or .1 for IMS start?  
IF OSVER=01          Check Version 1  
  IF OSREL=08         Check Release 8  
    S CICSnew  
  IF OSMOD=00         Check Mod 00  
    S IMS00  
  IF OSMOD=01         CHECK Mod 01  
    S IMS01  
  ENDIF  
ENDIF  
ENDIF  
ENDIF  
.....
```

IF RACFgrp | IF SECGRoup (Generics Supported)**=|EQ|^=|NE|>|GT|<|LT|/|GE|\|LE****RACFGroup****“OR”, “|”, “AND”, and “+” are supported in this command****“ORIF”, “||”, “ANDIF”, “++” supported by this command****IFSECGRoup**

Alias of IF SECGRoup See IF SECGROUP section for more information.

IF SECGROUP | IF RACFGROUP (Generics Supported)

=|EQ|^=|NE|>|GT|<|LT|/|GE|\\LE

RACFGROUP

“OR”, “|”, “AND”, and “+” are supported in this command

“ORIF”, “||”, “ANDIF”, “++” supported by this command

IFRACFGROUP

The IF RACFGROUP control command is used to cause the command script to test for the Task's RACF group of the issuer of the message being processed. You are required to identify a "group name" that you wish to test for. You may not specify more than one RACFGROUP at a time. This command begins an "IF NEST" of commands that will be done only IF the condition being tested for is a TRUE statement. If the RACFGROUP you have elected to test for is found to not be the actual execution RACFGROUP, the condition is set to FALSE, and all subsequent commands until the next ELSE or the matching ENDIF statement are bypassed. The ELSE statement will allow commands that should be executed on a FALSE condition to be executed, (i.e. the requested class is NOT the class you are testing for). Indentation of IF levels is not required, but makes the script more readable to a human.

Parameters:

RACFGROUP

Example:

```
.....
* See if we are a Payroll group job
IF RACFGROUP payroll
  Wto Payroll JOB &TASKNAME is running
ELSE
  WTO &taskname was not a payroll job because it's RACF group is &RACFGROUP
ENDIF
.....
```

IF StepCC(Stepname) or (Stepname.Procstep)

=, EQ, NE, >, GT, <, LT nnnn (4 character Maximum Condition code to check for)

IFMAXCC

The IF StepCC control command is used to cause the command script to test for the Task's Condition code of a particular step of the issuer of the message being processed. If the step doesn't exist or has not yet ended, the test will fail. You are required to identify a numeric or (in the case of an abend), alphanumeric code to check against. You may not specify more than one code at a time. This command begins an "IF NEST" of commands that will be done only IF the condition being tested for is a TRUE statement. If the StepCC code you have elected to test for is found to not be the actual execution StepCC of the step designated, the condition is set to FALSE, and all subsequent commands until the next ELSE or the matching ENDIF statement are bypassed. The ELSE statement will allow commands that should be executed on a FALSE condition to be executed, (i.e. the requested class is NOT the class you are testing for). Indentation of IF levels is not required, but makes the script more readable to a human.

Parameters:

(Stepname) or (Stepname.Procstep)

'=' or 'EQ' or 'NE' or '<' or 'LT' or '>' or 'GT'

Nnnn or xxxx (numeric or alphanumeric code)

Example:

* Do some code checking?

IF Taskname = CICS002

IF STEPCC(GO.STEP002) NE 0000

 **We have a problem

 WTO CICS002 failed in step GO.STEP002

 Email STEPCC ← this sends all of the STEPCC's

 TO: CICSSUPT ← to the nickname CICSSUPT

 From: CICS002@thissite.net

 Subject: &TASKNAME failed in step GO.STEP002 with &MAXCC

 MSG: We had some problem, so we are going to notify CICS support
 that there is something that might need to be done

 SENDMAIL

ELSE

 ** do nothing

ENDIF

ENDIF

IF STARTED | IF ACTIVE**(Generics Supported)**

=|eq|ne|^=
TASKNAME

“ORIF”, “||”, “ANDIF”, “++” supported by this command

IFSTARTED

The IF STARTED control command is used to cause the command script to test for a specific JOB or TASK to be running. This command begins an “IF NEST” of commands that will be done only IF the condition being tested for is a TRUE statement. If the JOB/TASK is found to be not running the condition is set to FALSE, and all subsequent commands until the next ELSE or the matching ENDIF statement are bypassed. The ELSE statement will allow commands that should be executed on a FALSE condition to be executed, (i.e. the requested TASKNAME is NOT active). Indentation of IF levels is not required, but makes the script more readable to a human.

Parameters:

taskname

Taskname - required - specifies the JOB or TASK that you wish to test for.

Example:

```

.....
* See if CICS001 is already running
IF STARTED CICS001                                <-----IF LEVEL 1 begin
* Shut it down if so
  F CICS001,CEMT P SHUT
*   See if it's still running
    IF STARTED CICS001                            <-----IF LEVEL 2 begin
      F CICS001,CEMT P SHUT IMM
*     See if it is STILL running
      IF STARTED CICS001                          <-----IF LEVEL 3 begin
        Cancel CICS001
* CICS is NOW not running so start the backup job
      F A,CICSBKUP
    ENDIF                                         <-----IF LEVEL 3 end
  ELSE                                           <-----IF LEVEL 2 ELSE
* CICS is NOW not running so start the backup job
    F A,CICSBKUP
  ENDIF                                         <-----IF LEVEL 2 end
ELSE                                           <-----IF LEVEL 1 ELSE
* CICS is not running so start the backup job
  F A,CICSBKUP
ENDIF                                           <-----IF LEVEL 1 end
.....

```

IF STOPPED | IF INACTIVE (Generics Supported)

**=|eq|ne|^=
TASKNAME**

“ORIF”, “||”, “ANDIF”, “++” supported by this command

IFSTOPPED

The IF STOPPED control command is used to cause the command script to test for a specific JOB or TASK to be **NOT** running. This command begins an “IF NEST” of commands that will be done only IF the condition being tested for is a TRUE statement. If the JOB/TASK is found to be ACTIVE, the condition is set to FALSE, and all subsequent commands until the next ELSE or the matching ENDIF statement are bypassed. The ELSE statement will allow commands that should be executed on a FALSE condition to be executed, (i.e. the requested TASKNAME is RUNNING/ACTIVE. Indentation of IF levels is not required, but makes the script more readable to a human.

Parameters:

taskname

Taskname - required - specifies the JOB or TASK that you wish to test for.

Example:

* See if CICS001 is not running

IF STOPPED CICS001

<-----IF LEVEL 1 begin

* Yes, not running so start backups

F A,CICSBKUP

ELSE

<-----IF LEVEL 1 ELSE

* Running so shut it down

F CICS001,CEMT P SHUT

* See if it's down yet

IF STOPPED CICS001

<-----IF LEVEL 2 begin

* no longer running so start backups

F A,CICSBKUP

ELSE

<-----IF LEVEL 2 ELSE

F CICS001,CEMT P SHUT IMM

* See if it is STILL running

IF STARTED CICS001

<-----IF LEVEL 3 begin

Cancel CICS001

* CICS is NOW not running so start the backup job

F A,CICSBKUP

ENDIF

<-----IF LEVEL 3 end

ENDIF

<-----IF LEVEL 2 end

ENDIF

<-----IF LEVEL 1 end

.....

IF STRING

xx(yy) text

IFSTRING

The IF STRING control command is used to cause the command script to test for a specific STRING to be contained at a specific offset (xx) for a specific length(yy) within the MPF message being processed. The offset number (xx) must be specified as a 2-digit numeric relative to 00 (zero) and 00 is the first character of the messageID. The Length can be any 2-digit numeric length (up to 64 characters). Do not enclose the text to be compared in quotes, the comparison starts with the first character/space following the first blank after the ")" (right parentheses of the xx(yy) parameter. The first character of any message (zero (00) is ALWAYS the first character of the message ID itself. For instance, see the following message are identified below the message (Case sensitive):

```
SYZSP21E - Multiple Blank Lines were skipped.  
0          1          2          3          4          5  
012345678901234567890123456789012345678901234567890
```

```
Offset 10 is "-"  
Offset 20 is "e"
```

```
STRING 22(5) is "Blank"  
STRING 28(15) is "Lines were skip"
```

This command begins an "IF NEST" of commands that will be done only IF the condition being tested for is a TRUE statement. If the STRING of the length specified is found to be not be at the location tested, the condition is set to FALSE, and all subsequent commands until the next ELSE or the matching ENDIF statement are bypassed. The ELSE statement will allow commands that should be executed on a FALSE condition to be executed, (i.e. the requested STRING is NOT found). Indentation of IF levels is not required, but makes the script more readable to a human.

Parameters:

nn text

xx - required - the STRING offset (beginning with zero) of the text to be tested for.
(yy)- required - the STRING length of the text to be tested for.
Text - required - The actual TEXT of the word at the offset nn to be checked.

Example:

```
.....  
* Test SYZSP21E message for the string "Blank Lines"  
IF STRING 22(11) Blank Lines  
* If just Blank lines, it's okay  
* do nothing  
  ELSE  
* non-blank lines could be are a problem so page Tech Support  
  S PAGE,USER=TECHSUPP,SEV=4,ERROR="SyzSP21E - SyzSpool can't process a task"  
ENDIF  
.....
```

IF SUBMETHod (Generics Supported)

=|eq|ne|^=

Submit method source (INTRDR, RDRn, STCINRDR, etc.)

“OR”, “|”, “AND”, and “+” are supported in this command

“ORIF”, “||”, “ANDIF”, “++” supported by this command

IFSUBMETHOD

The IF SUBMETHod control command is used to cause the command script to test for the Task's Submit method (where the task entered the system) of the issuer of the message being processed. You are required to identify a valid source that you wish to test for. You may not specify more than one source at a time. This command begins an “IF NEST” of commands that will be done only IF the condition being tested for is a TRUE statement. If the SUBMIT method you have elected to test for is found to not be the actual execution SUBMIT method, the condition is set to FALSE, and all subsequent commands until the next ELSE or the matching ENDIF statement are bypassed. The ELSE statement will allow commands that should be executed on a FALSE condition to be executed, (i.e. the requested class is NOT the class you are testing for). Indentation of IF levels is not required, but makes the script more readable to a human.

Parameters:

Name

Example:

```
.....
* See if we are a dealing with jobs from Remote 3
IF SUBMETH = R3.RDR1
  Wto The job came from remote 3
  .... (other commands)
ELSE
  WTO &taskname was not submitted from Houston, submit method=&SUBMITMETHOD
ENDIF
.....
```

IF SYSID (Generics Supported)

=|EQ|^|=|NE|>|GT|<|LT|/|GE|\\LE
SYSTEMID

The '=' is not required in this request and can be eliminated if desired
"OR", "|", "AND", and "+" are supported in this command
"ORIF", "||", "ANDIF", "++" supported by this command

IFSYSID

The IF SYSID control command is used to cause the command script to test for the SYSTEM ID (or generic ID) that the script processor is executing under. You are required to identify the SYSTEMID that you wish to test for. You may not specify more than one SYSTEMID at a time.

This command begins an "IF NEST" of commands that will be done only IF the condition being tested for is a TRUE statement. If the SYSTEMID you have elected to test for is found to not be the current SYSTEM NAME, the condition is set to FALSE, and all subsequent commands until the next ELSE or the matching ENDIF statement are bypassed. The ELSE statement will allow commands that should be executed on a FALSE condition to be executed, (i.e. the requested SYSTEM NAME is NOT the one you are testing for). Indentation of IF levels is not required, but makes the script more readable to a human.

Parameters:

SYSTEMID

The SYSTEMID may contain the following Generic Characters:

'%' means any single character is to be considered a match
'*' means from that point on is considered a match

The one to 8 character SYSTEM name (as specified in parmlib) is required.

Example:

```
.....  
* Make sure it's the PRODUCTION system  
IF SYSID A7PROD  
  $P PRT15  
  $PI  
  $TI1-5,ABCD  
  $SI1-5  
ELSE
```

```
* not PRODUCTION, so must be TEST
$PI
$TI1-5,CDEF
$SI1-5
ENDIF
.....
```

IF SYSPLEX (Generics Supported)

=|eq|ne|^=

SYSPLEXNAME

The '=' is not required in this request and can be eliminated if desired

"OR", "|", "AND", and "+" are supported in this command

"ORIF", "||", "ANDIF", "++" supported by this command

IFSYSPLEX

The IF SYSPLEX control command is used to cause the command script to test for the SYSPLEX NAME (or generic name) that the script processor is executing under. You are required to identify the SYSPLEX NAME that you wish to test for. You may not specify more than one SYSPLEX NAME at a time.

This command begins an "IF NEST" of commands that will be done only IF the condition being tested for is a TRUE statement. If the SYSPLEX NAME you have elected to test for is found to not be the current SYSPLEX NAME, the condition is set to FALSE, and all subsequent commands until the next ELSE or the matching ENDIF statement are bypassed. The ELSE statement will allow commands that should be executed on a FALSE condition to be executed, (i.e. the requested SYSPLEX NAME is NOT the one you are testing for). Indentation of IF levels is not required, but makes the script more readable to a human.

Parameters:

SYSPLEXNAME

The SYSPLEXname may contain the following Generic Characters:

'%' means any single character is to be considered a match

'*' means from that point on is considered a match

The one to 8 character SYSPLEX name is required.

Example:

```
.....
* Make sure it's the PRODUCTION sysplex
IF SYSPLEX PRODPLEX
  $P PRT15
  $PI
  $TI1-5,ABCD
  $SI1-5
ENDIF
.....
```

TASKOWNER | IF USERID (Generics Supported)

=|EQ|^=|NE|>|GT|<|LT|/|GE|\|LE

UserID (up to 8 characters)

The '=' is not required in this request and can be eliminated if desired

“OR”, “|”, “AND”, and “+” are supported in this command

“ORIF”, “||”, “ANDIF”, “++” supported by this command

IFTASKOWNER

Alias of IF USERid, see IFUSERID for more information

IF TIME | IF CURRENTTIME (Generics Supported)

=|EQ|^|=|NE|>|GT|<|LT|/|GE|\\LE the “current” time **HH:MM**
“OR”, “|”, “AND”, and “+” are supported in this command
“ORIF”, “||”, “ANDIF”, “++” supported by this command

IFTIME

The IF TIME control command is used to cause the command script to test for the current, time of day and compare the current time to the value specified. Generics are allowed, but they should not be used, or used carefully with Greater/Less time. You are required to identify the Time of Day in HH (Hours) and MM (minute) format that you wish to test for. You may not specify more than one specific time. But you can specify more than one time per line via the OR or ORIF parameter(s). This command begins an “IF NEST” of commands that will be done only IF the condition being tested for is a TRUE statement. If the current Time does not correctly compare to the time you have selected, the condition is set to FALSE, and all subsequent commands until the next ELSE or the matching ENDIF statement are bypassed. The ELSE statement will allow commands that should be executed on a FALSE condition to be executed, (i.e. the current time fails the compare to the requested time of day you have specified in the control command). Indentation of IF levels is not required, but makes the script more readable to a human.

Parameters:**HH:MM**

Time of day in 24 hour format 00:00 through 23:59.

Example:

```
.....
* Make sure it's before 8am or after 5pm
IF TIME > 07:59 ++ IF TIME < 17:01
  $P PRT15
  $PI
  $TI1-5,ABCD
  $SI1-5
ELSE    ←- else it is before 8am but after 5pm
  $PI
  $TI1-5,CDEF
  $SI1-5
ENDIF
.....
```

IF USERID | TASKOWNER (Generics Supported)

=|EQ|^=|NE|>|GT|<|LT|/|GE|\\LE

UserID (up to 8 characters)

The '=' is not required in this request and can be eliminated if desired

"OR", "|", "AND", and "+" are supported in this command

"ORIF", "||", "ANDIF", "++" supported by this command

IFUSERID

The IF USERID control command is used to cause the command script to test for the specific (or generic) JOB/STC or TSU OWNER (RACF/ACF2, etc.) ID of the task that issued the message that caused this script to be processed. This command begins an "IF NEST" of commands that will be done only IF the condition being tested for is a TRUE statement. If the USERID you have elected to test for is found to not be the USERID that issued the message that this script is processing, the condition is set to FALSE, and all subsequent commands until the next ELSE or the matching ENDIF statement are bypassed. The ELSE statement will allow commands that should be executed on a FALSE condition to be executed, (i.e. the requested USERID is NOT the one you are testing for). Indentation of IF levels is not required, but makes the script more readable to a human.

Parameters:

UserID (up to 8 characters)

The UserID may contain the following Generic Characters:

'%' means any single character is to be considered a match

'*' means from that point on is considered a match

The one to 8 character TASK NAME is required.

Example:

.....For message DFHSI1517

IF USERID CICS*

* if CICS123 start DB2 Connect A

IF **MSGTASK** CICS123

S DB2CONA

ENDIF

ENDIF

.....

IF WORD (Generics Supported "*" only)

nn text | text* (generics allowed)

IFWORD

The IF WORD control command is used to cause the command script to test for a specific WORD (or a generic word of at least 1 byte followed by an asterisk "*") to be contained at a specific WORD offset within the MPF message being processed. "WORD"s are delimited by at least one blank space. The WORD number must be specified as a 2-digit numeric relative to 00 (zero). The first WORD or any message (WORD zero (00) is ALWAYS the message ID itself. For instance, the 7 words of the following message are identified below the message (Case sensitive):

```
SYZSP21E - Multiple Blank Lines were skipped.  
  (00)   (01)  (02)   (03) (04)  (05)  (06)  
Word 00 = SYZSP21E  
Word 01 = - (hyphen)  
Word 02 = Multiple  
Word 03 = Blank  
Word 04 = Lines  
Word 05 = were  
Word 06 = Skipped
```

This command begins an "IF NEST" of commands that will be done only IF the condition being tested for is a TRUE statement. If the WORD (or generic word if specified) is found to be not be at the location tested, the condition is set to FALSE, and all subsequent commands until the next ELSE or the matching ENDIF statement are bypassed. The ELSE statement will allow commands that should be executed on a FALSE condition to be executed, (i.e. the requested WORD is NOT found). Indentation of IF levels is not required, but makes the script more readable to a human.

Parameters:

nn text (text may be generic with * only, not % at this time)
nn - required - the WORD offset (beginning with zero) of the word to be tested.
Text - required - The actual TEXT of the word at the offset nn to be checked.

Example:

```
.....  
* Test SYZSP21E message word (03) for the text "Blank"  
IF WORD 03 Blank  
    * If just Blank lines, it's okay  
    * do nothing
```

```
ELSE
    * non-blank lines could be are a problem so page Tech Support
    S PAGE,USER=TECHSUPP,SEV=4,ERROR="SyzSP21E - SyzSpool can't process
    a task"
ENDIF

*See if Word 4 starts with "Li" (for "Lines")
IF Word 04 Li*
    Do something...
ELSE
    Do something else
ENDIF
.....
```

IF WORDS (Generics Supported)

WORD1 WORD2 WORD3 WORD4

IFWORDS

The IF WORDS control command is used to cause the command script to test for a specific SET of (up to 4) WORDS to be contained within the MPF message being processed. "WORD"s are delimited by at least one blank space and are case sensitive. The individual words to be checked can occur in ANY order, but they all MUST occur at least once within the message.

Each individual word can be a generic i.e. SYZ* for all words that start with SYZ and SYZ%%% for all words that start with SYZ but are at least 6 characters long. You cannot put a "*" in the middle of a word so SYZ*123 is not valid, but SYZ%%123 would be valid for any word that starts with SYZ and ends with 123 that is 8 characters long no matter what the two middle characters are, so SYZAB123 and SYZXX123 are both valid for SYZ%%123.

This command begins an "IF NEST" of commands that will be done only IF the condition being tested for is a TRUE statement. If the WORDS are ALL NOT found to be in the message, the condition is set to FALSE, and all subsequent commands until the next ELSE or the matching ENDIF statement are bypassed. The ELSE statement will allow commands that should be executed on a FALSE condition to be executed, (i.e. one or more of the requested WORDS is NOT found). Indentation of IF levels is not required, but makes the script more readable to a human.

Parameters:

text
WORD - required (up to 4 separate words) - the WORD(s) (in any order) of the message to be tested for.

Example:

```
.....  
* Test SYZSP18F message for "FACILITY" "MAKER" "COMPLETE"  
IF WORDS Facility Maker Complete  
F Syzspool,A=12H * make Syzspool delay processing of output until 12 hours old  
* page Tech Support  
S PAGE,USER=TECHSUPP,SEV=1,ERROR="SyzSP18F - SyzSpool RACF facility error"  
ENDIF  
.....
```

IF WORKLOAD (Generics Supported)

=|EQ|^=|NE|>|GT|<|LT|/|GE|\|LE

Workload Name this task is using

“OR”, “|”, “AND”, and “+” are supported in this command

“ORIF”, “||”, “ANDIF”, “++” supported by this command

IFWORKLOAD

The IF WORKLOAD control command is used to cause the command script to test for the Task's WORKLOAD name setting (this is the WLM setting for this task) of the issuer of the message being processed. You are required to identify a "workload name" that you wish to test for. You may not specify more than one WORKLOAD NAME at a time. This command begins an "IF NEST" of commands that will be done only IF the condition being tested for is a TRUE statement. If the WORKLOAD NAME you have elected to test for is found to not be the actual execution WORKLOAD NAME, the condition is set to FALSE, and all subsequent commands until the next ELSE or the matching ENDIF statement are bypassed. The ELSE statement will allow commands that should be executed on a FALSE condition to be executed, (i.e. the requested class is NOT the class you are testing for). Indentation of IF levels is not required, but makes the script more readable to a human.

Parameters:

WORKLOADName

Example:

```
.....
* See if we are a Production Batch group job
IF WORKload = PRODBAT
  Wto Production! JOB &TASKNAME is running
  ..... (other commands)
ELSE
  WTO &taskname was not a production job because it has
  WORKLOAD=&workload
ENDIF
.....
```

LOWLIGHT

The LOWLIGHT control command is used to cause the SyzMPF/z scripting language to make highlighted messages no longer highlighted on the operator console. This command can also be issued in the "AUTO(LOWLIGHT)" part of the actual MPFLSTxx member, thereby negating the need to have a corresponding SyzMPF/z MPF member to be processed

Parameters:

NONE

Example:

In MPFLST MEMBER:

```
.....  
DFHSM013*,SUP(NO),USEREXIT(SYZMPFZ),AUTO(LOWLITE)  
.....
```

OR in a SyzMPFz MPF script for the same DFHSM013* series of messages

```
.....  
* Un-highlight the messages  
LOWLIGHT
```

```
.....
```

MSGCOLOR

color or **videomode**

The MSGCOLOR control command is used to cause the SyzMPF/z scripting language to set the MPF message being processed to be issued in a specific color (red, white, blue, green, yellow, pink, turquoise) or video mode (Underline, blink, reverse video). You may combine these options (i.e. RED and reverse Video), but they must occur on separate lines, you cannot put both commands on the same lines. The colors are overriding, in that if you specify MSGCOLOR RED to change it to RED and then MSGCOLOR BLUE, the message will be issued in BLUE, not RED and BLUE.

Parameters:

color - RED, WHItE, BLUe, GREen, YELLow, PINK, TURquoise)
videomode - UNDErline, BLInk, REVvideo, LOWlight

Example:

.....
* Make the message Pink, and reverse video

MSGCOLOR PIN
MSGCOLOR REV

.....

NOCONSOLE

(use the **SUPPRESS** command or read below)

NOCONSOLE is not a SyzMPF/z command, but seems like a good place to explain how to **NOT** display a message on the operator console without even using SyzMPF/z, but still allow it to be displayed in the Hardcopy SYSLOG and in the JOBLLOG of the job that issued the message. To do this the following trick will suffice.

In the MPFLST member of parmlib you specify the message you want to suppress only from the console and the ",SUP(YES)" parameter. For instance, if you want to suppress the following message:

DUMBMSG11 This is a really dumb message.

You would code the entry in MPFLSTxx of Parmlib as follows:

DUMBMSG11,SUP(YES)

That's it! The message will **still** go to the SYSLOG and the JOBLLOG of the job that issued it and no longer be sent to the operator's console.

Parameters:

NONE

Example:

In MPFLST MEMBER:

```
.....  
$HASP373,SUP(YES)          (JOB Started)  
.....
```

NOJOBLOG

The NOJOBLOG control command is used to cause the SyzMPF/z scripting language to SUPPRESS the MPF message being processed from being recorded on the JOBLOG of the task that issued the message. This command can also be issued in the "AUTO(NOJOBLOG) part of the actual MPFLSTxx member, thereby negating the need to have a corresponding SyzMPF/z MPF member to be processed.

Parameters:

NONE

Example:

In MPFLST MEMBER:

```
.....  
$HASP373,AUTO(NOSYSLOG)      (JOB Started)  
.....
```

OR in a SyzMPFz MPF script for the same \$HASP373 messages

```
.....  
* completely suppress the messages from the user's JOBLOG  
NOJOBLOG  
  
.....
```


NOSYSLOG

The NOSYSLOG control command is used to cause the SyzMPF/z scripting language to SUPPRESS the MPF message being processed from being recorded on the system hardcopy log (SYSLOG). This command can also be issued in the "AUTO(NOSYSLOG)" part of the actual MPFLSTxx member, thereby negating the need to have a corresponding SyzMPF/z MPF member to be processed.

Parameters:

NONE

Example:

In MPFLST MEMBER:

```
.....  
$HASP309,AUTO(NOSYSLOG)  
.....
```

OR in a SyzMPFz MPF script for the same \$HASP309 messages

```
.....  
* completely suppress the messages from the hardcopy log  
NOSYSLOG  
  
.....
```

PARSE2words

***** As of version 5.5, this parameter is no longer absolutely necessary, it is performed automatically if the script asks for a &wnn element if PARSE2WORDS was not already performed. However if you use the PARSEDELIMITER commands you can use PARSE2WORDS to “Reparse” the message into “new” words that use a delimiter “other than” blank**

The PARSE2WORDS control command is used to cause SyzMPF/z to “parse” the MPF message currently being processed and make each one (or all) individually available to be used in any command to be issued. The individual “words” of the message text are moved to variables &W00 (zero-zero) which is the normally the MessageID, through &W49. That allows the first 50 words of the message text to be individually addressable in any command the site wishes to issue. Previously, (before Version 2 of SyzMPF/z) only one word at a time was available (two if you combined GET WORD and GET STRING). Now up to the first 50 words of the message text are able to be used.

See & command for more information in using the output from this command.

There are no IF/THEN/ELSE requirements for this script command, it parses the words of the MPF message and optionally (if the ECHO=PARSE or ECHO=ALL option is active), will display each of the words on the console to allow the site to more easily build command scripts .

Parameters:

n/a

Example: Message is → ABC1234I This is a test of the parse2 command

.....

WTO &W07 &W08 &W04 &W05 &W00

**** Results in the following WTO:

“Parse command test of ABC1234I”

.....

PARSEDELIMITER

b

The PARSEDelimiter command will allow you to change the default parse delimiter from BLANK to some other character. This is useful If you wish to parse the message you are processing in the script by something other than BLANKs.

Parameters:

B - any character, the default is a BLANK

Example: Message is → ABC1234I This is a test. Of the parse2 command.

.....

WTO &W07 &W08 &W04 &W05 &W00

**** Results in the following WTO:

"Parse command. test. of ABC1234I"

PARSEDELIMITER . ← using a period

WTO &W1 &W0

**** Results in the following WTO:

**** &w1 &W0

" Of the parse2 command ABC1234I This is a test " ← the periods are removed

.....

QUERY

DETAILS | DEFAULTS | WORDS

The QUERY control command is used to cause SyzMPF/z to send either the current GLOBAL defaults or the DETAILS of every script (name, number of times processed, last time processed, etc.) to the console.

Parameters:

DETAILS	-	show the detail script statistics
DEFAULTS	-	show the Global defaults
WORDS	-	show the message words

Example: This can be part of any messages processing but “@” will be used

.....

***Script named “@”**

ECHO NONE

IFWORD 00 @QUERY

IFWORD 01 DEFAULTS

WTO '-----'

QUERY DEFAULTS

WTO '-----'

(EXIT)

ENDIF

IFWORD 01 DETAIL

WTO '-----'

QUERY DETAIL

WTO '-----'

(EXIT)

ENDIF

IFWORD 01 ALL

WTO '-----'

QUERY DEFAULTS

WTO '-----'

WTO '-----'

QUERY DETAIL

WTO '-----'

(EXIT)

ENDIF

ENDIF

***Script named “IEE305I”**

IFWORD 01 @*

← all “@anything” will not have message generated

SUPPRESS

(EXIT)
ENDIF

.....

*MPFLSTnn setting for making user commands (we use "@")
 @*,SUP(NO),USEREXIT(SYZMPFZ),AUTO(@) ← our @anything commands
 IEE305I,SUP(NO),USEREXIT(SYZMPFZ) ← to avoid invalid command messages

If operator types "@QUERY DEFAULTS", the results are:

```
'-----'
SyzMPF/z loaded/refreshed on SUNDAY APRIL 27, 2014 at: 15:09:52.
SyzMPF/z will expire in 271 days.
Simluate mode : No
Quiet mode   : No
Debug mode   : Yes
Statistics    : Off
SyzMail Active: Yes
Max Scripts   : 99
Max Email Msg : 50
Pre-Script    : @FIRST
Post-Script   :
Customer      : 00-0000S Syzygy Inc. Main Site
Script Library: SYS1.SYZMPFZ.SCRIPTS
ECSA MemoryLoc: (1106D000) Len(00068000)
TimeZone      : -1000
System ID     : Z21
Sysplex Info  : Name:LOCAL ID:1A
LPAR name     : HERCULES
CPU Info      : 2098 ID:C01F
Email To: Products@SyzygyInc.com
Email From: HAD@SyzygyInc.com
Email Cc: NobodyCC@SyzygyInc.com
Email Bcc: NobodyBCC@SyzygyInc.com
Email ReplyTo: WebSiteSupport@SyzygyInc.com
Email Subject: Message from SyzMPF/z
'-----'
```

If operator types "@QUERY DETAILS", the results are:

```
'-----'
SCRIPT: $$$$DFLT      COUNT:   0 L-USED:  : :  / /
SCRIPT: $HASP39X      COUNT:   0 L-USED:  : :  / /
SCRIPT: $HASP395      COUNT:   3 L-USED: 14:56:42 04/27/14
SCRIPT: ###          COUNT:   0 L-USED:  : :  / /
```

SCRIPT: #####	COUNT: 3 L-USED: 12:56:11 04/27/14
SCRIPT: ###2	COUNT: 0 L-USED: : : / /
SCRIPT: ###3	COUNT: 0 L-USED: : : / /
SCRIPT: @	COUNT: 8 L-USED: 14:56:56 04/27/14
SCRIPT: @@	COUNT: 0 L-USED: : : / /
SCRIPT: @@@	COUNT: 0 L-USED: : : / /
SCRIPT: @FIRST	COUNT: 0 L-USED: : : / /
SCRIPT: @HASP395	COUNT: 0 L-USED: : : / /
SCRIPT: AOP075I	COUNT: 0 L-USED: : : / /
SCRIPT: CMDFIRST	COUNT: 0 L-USED: : : / /
SCRIPT: COF536I	COUNT: 0 L-USED: : : / /
SCRIPT: DFLT\$\$\$\$	COUNT: 0 L-USED: : : / /
SCRIPT: ERB100I	COUNT: 0 L-USED: : : / /
SCRIPT: ESP	COUNT: 0 L-USED: : : / /
SCRIPT: EZZ3250I	COUNT: 0 L-USED: : : / /
SCRIPT: FIRST@	COUNT: 0 L-USED: : : / /
SCRIPT: FREDREP	COUNT: 0 L-USED: : : / /
SCRIPT: FREDREPL	COUNT: 0 L-USED: : : / /
SCRIPT: ICH408I	COUNT: 0 L-USED: : : / /
SCRIPT: IEA989I	COUNT: 0 L-USED: : : / /
SCRIPT: IEE305I	COUNT: 8 L-USED: 14:56:42 04/27/14
SCRIPT: IEE362A	COUNT: 0 L-USED: : : / /
SCRIPT: IEFC452I	COUNT: 0 L-USED: : : / /
SCRIPT: IEF238D	COUNT: 0 L-USED: : : / /
SCRIPT: IEF403I	COUNT: 3 L-USED: 11:22:42 04/27/14
SCRIPT: IEF403I9	COUNT: 0 L-USED: : : / /
SCRIPT: IEF404I	COUNT: 0 L-USED: : : / /

....etc

'-----'

REPLY

```
##      ###      or #####,text
nn      nnn      or nnnn,text
```

REPLY

The REPLY control command is used to cause the command script to set a issue a reply to the WTOR MPF message being processed. This is the "standard" method of replying to WTOR's and is the simplest to use. The "nn", "nnn" or "nnnn" part of the command will depend on whether your system has RMAX of 2, 3 or 4 character REPLY IS's. You can always use 4, and it will be handled correctly, most sites use RMAX set to 2 (the default which has a maximum ID of 99), but many sites find that setting RMAX higher is convenient for them. The processing path method for nn is much shorter than for nnn or nnnn, and it is provided as a option solely because of that shorter path length.



*note: For single or simple replies, you can use the MPFLSTnn message line AUTO(#xxxxx...) support as outlined in the "QuickReplies MPFLSTxx AUTO(#xxxx...) actions which can be taken with SyzMPF/z" section earlier in this manual.

There are no IF/THEN/ELSE requirements for this script command, it issues the REPLY to the WTOR with the text as specified. The "nn", "nnn" or "nnnn" is replaced with the ACTUAL REPLY ID of the WTOR being processed.

Parameters:

```
## (or) nn
### (or) nnn
#### (or) nnnn,text
```

n (or) #- **a placeholder to show where to place the WTOR ID to be replied to.**
Text - **the text of the reply to be issued**

Example:

```
.....
*      Reply to the WTOR with "COLD"
```

REPLY NNNN,COLD (results in WTOR nn,COLD where 'nn' is the WTOR ID)

```
.....
```

SKIPTO

MPF-Script-Name

The SKIPTO control command is used to cause SyzMPF/z to load another MPF Scripting member and process it as a continuation of the current member. This can be handy on those rare instances when 100 lines of command script is not enough to perform all that you want to do, or when the site might want to perform some static function for several different messages which are exactly the same and they don't want to duplicate it in to each script member. When you "skip" to the new MPF script, the settings and original message text from the message that started the original script are still the same, it's just like we have continued the script without stopping only we have loaded a new MPF script to process that is named by the member Script-Name supplied.

Parameters:

MPF-Script-Name

Up to 8 characters that match the name of a valid MPF script in the MPF script library.

Example:

```
.....For message DFHSI1517
IF MSGID DFHSI1517
* if CICS123 start DB2 Connect A
  IF MSGTASK CICS123
    S DB2CONA
  ENDIF
* if CICS456 start DB2 Connect B
  IF MSGTASK CICS456
    S DB2CONB
  ENDIF
ENDIF
* no matter what CICS this is, we want to send a page contained in "PAGETS"
SKIPTO PAGETS
.....
...
```


STRIP

WORD (ABCD)

Or

STRing (ABCD)

The STRIP control command is used to cause SyzMPF/z to take an existing WORD or STRING (depending on the command parms) and REMOVING all occurrences of the characters or numbers or special characters that are contained within the parentheses. You can use this command recursively if there are more than 4 characters that you wish to remove from a already set WORD or STRING.

For instance if the message were to be:

ABC012345 This is a test message

And you were to have already coded a "GET WORD 4" command, then the &WORD variable would be set to 'message'.

If you were to code: STRIP WORD (esa)

Then we would remove all occurrences of the letters 'e', 's' and 'a' from the WORD and we end up with WORD being set to 'mg' because all of the e's, s's and a's have been removed.

Currently you cannot remove ")" (right parentheses) from a WORD or STRING but that will be changed in a later release.

Parameters:

WORD or STRING

Depending on whether you are changing a previously obtained &WORD or &STRING

(1234)

Up to 4 characters, numbers or special characters to be removed from the WORD or STRING

Example:

.....For message ABC012345 This is a test message

GET WORD 4

STRIP WORD (aes) ← removes all a, e and s from the WORD

WTO &WORD ← results in "MG" being displayed on the console

.....

...

SUPPRESS

The SUPPRESS control command is used to cause the SyzMPF/z scripting language to SUPPRESS the MPF message being processed from being recorded on the Operator Console, the SYSLOG or the JOBLOG of the task. This command can also be issued in the "AUTO(SUPPRESS) part of the actual MPFLSTxx member, thereby negating the need to have a corresponding SyzMPF/z MPF member to be processed.

****NOTE:** Care should be taken when using this script command because it suppresses the message totally. It's like it never happened and can make things difficult to find if you are trying to debug a problem and someone has suppressed all of the messages you need. You should only suppress messages that you REALLY do not want to have recorded ANYWHERE.

Parameters:

NONE

Example:

In MPFLST MEMBER:

```
.....  
$HASP309,AUTO(SUPPRESS)  
.....
```

OR in a SyzMPFz MPF script for the same \$HASP309 messages

```
.....  
* completely suppress the messages  
SUPPRESS
```

```
.....
```

TimeZone or TZ =

Time zone offset to use (default = PST)

The "TimeZone or TZ" control command is used to tell SyzMPF/z what Time Zone to use for all outgoing eMail messages. This setting is not required but is suggested because the default is PST. Any valid offset may be used, including the standard US time zone offsets, PST,PDT,EDT,EST,CST,CDT,etc.) The site can also specify the offsets to the west or (- negative) or east (+positive) of GMT. i.e. -0700 is the pacific time zone (PST) west 7 hrs.

Parameters:

Timezone or offset (+/-) Default is PST (TZ=-0700)

Example:

```
.....  
TimeZone = -0700      ← Pacific time (PST)  
.....
```

SYZMAIL =**Yes | No**

The “SYZMAIL” control command is used to tell SyzMPF/z whether or not the SYZMAIL/z product is installed at this. This may have been set by the startup parameter in parmlib, but can be overridden for an individual script.

Parameters:**Yes | No (NO is the default)****Example:**

.....

SyzMail = Yes

.....

WTO

anything

The WTO control command is used to issue informational messages to the operators console, there are no restrictions on the contents, which are sent exactly as they are written.

Parameters:

anything or nothing

Up to 72 characters are sent following the WTO control parameter.

Example:

```
.....  
WTO  
WTO *****  
WTO * *  
WTO * This is a test *  
WTO Asterisks are not necessary *  
WTO * *  
WTO *****  
.....
```

WTOH

anything

STICKY

The WTOH control command is used to issue highlighted informational messages to the operators console, there are no restrictions on the contents, which are sent exactly as they are written. The message(s) are sent as highlighted and non-rollable.

Parameters:

anything or nothing

Up to 72 characters are sent following the WTOH control parameter.

Example:

```
.....  
WTOH Hey!, WAKE UP!!!!  
.....
```