Syzygy Incorporated

[Version 10.0.0]
August 21, 2022

SyzCMD/z

[Installation and User’s Guide]

Complete on-demand script-based system automation facility

Revision History

Version	Date	Revision Description
10.0	8/20/2022	Added &LDOW command to give long day of week name, i.e. “Monday” instead of the “MON” that &DOW gives. Added ability to create Dynamic variables and initialize them. DVAR command to initialize a Dynamic variable Dynamic variables are denoted with “#” (built-in variables still begin with “&”) Dynamic variable name can be up to 10 characters Dynamic Variable length can be up to 50 characters as a string containing any characters or even blank spaces. Dynamic Variables can contain Symbolic or internal variable names and static text. It’s possible to create more than 50 variables upon request Dynamic Variables can be used just like static and built-in variables Dynamic Variables can contain spaces and special characters Dynamic Variables can be cleared or deleted Dynamic Variables can be checked for their values or for if they merely exist (yet). Commands can be continued to a second line, and now each line’s limit is 126 characters, (specified across two lines). The continuation character is any character in column “72” and the continuation starts in the first character of the next line. Maximum WTO or WTOR message is now 111 characters. Use of variables may make the length up to 126 characters. If debugging is turned on, the TRUE|FALSE|IFNEST are now presented in a common offset of the message (far right). Variables may be set and used via SubStrings that specify the begin point and the length of the variable to use. Log dataset use has been enhanced to allow any size dataset to be used. Corrected the WTO, WTOH, STICKY and WTOR commands to not automatically uppercase the text. Many updates to manual. Added more samples. Update WAIT UNTIL STOPPED to have same MaxWaitTime as WAIT UNTIL STARTED

		Added WTOPREFix command and startup parm to denote whether the site wants the WTO and WTOH command messages to be prefixed with the "* taskname *" prefix. The default is YES Added ABEND processing to create any type of ABEND (ABEND processing is for testing specific ABEND codes It is not meant to be used for normal processing.)
9.0.4	11/23/2021	PAUSE, WAIT and DELAY are now fully interchangeable, you can use any of the 3 names to provide the same functions. Added MAXWAIT ability to "WAIT UNTIL STARTED" command via enclosing the time to wait in parentheses: WAIT UNTIL STARTED **(maxwait via 9999H|M|S)** 9999H|M|S If the MAXWAIT is hit, then the "IF EXPIRED" variable is set to TRUE|YES. The update places two sets of times on the command line, the one in parentheses should normally go first, and it is the maximum time that the wait until started process will wait for the named task to start. It can be specified in 9999 Hours, Minutes or Seconds. This parm is followed by the optional amount of time that the task must be actually in operation to qualify. For instance, if the task in question is CICSA, and it must be up for at least a continuous 30 seconds to continue with the process, but you don't want to wait more than 5 minutes for it to be actually started for that 30 seconds, then you would code: WAIT UNTIL STARTED CICSA (5m) 30s If the process is as desired, i.e. CICSA is started and runs continuously for 30 seconds (not just started 4 times over the course of 30 seconds), then the wait is ended and the script continues with no settings changed. If 5 minutes goes by, then the MAXWAIT of 5 minutes expires and the "IF EXPIRED" variable is set to TRUE or YES. To utilize this in a script, a simple example is as follows, in this case we wait for CICS to be started for 30 continuous seconds, but something happened to CICA, so it did not start within the 5 minute "limit" that we set. WAIT UNTIL STARTED CICSA (5m) 30s IF EXPIRED • Yes it is expired *we have to do something about the fact that it's not running. WTOH CICSA dis not start within the 5 minute time limit …DO something about it ELSE • No, it started fine and ran for at least 30 secs

		*nothing to do WTOH CICSA started okay ENDIF
9.0.3	05/10/2021	MAXWAIT=9999h|m|s now valid for all commands that perform a "wait" for something. STOPWAIT is now a valid alternate name for STOPCHECK Added "ALL" parm to UNSTICK. This will allow "unsticking" all messages generated by STICKY instead of just the last one.
9.0.2	11/23/2020	All wait related messages will now contain the "modify name" in parentheses within the wait message. For instance, if the script task is a started task, the name provided is the name which is to be used to interact with the task. If the task is a batch job, then the name that Is provided is the jobname (which is how you modify a batch job's script). For instance: **S SHUTDOWN.IPL** will start a task called shutdown, with a STCID of "**IPL**". You would interact with the task by the STCID "**IPL**" and not the task name "**SHUTDOWN**". If you don't provide the STCID, then z/OS automatically uses the hexadecimal unit name of the dasd volume which contains the script. **<u>This is an IBM restriction</u>**. So, if the script that the SHUTDOWN started task is executing is located in a dataset called "SYS1.COMMANDS" on volume DASD01 which is located on unit address x'1123', then the STCID will be "1123" and you would interact with the SHUTDOWN task via: **"F 1123,command".** Again, this is and has always been an IBM restriction. Enhancements to **ASKOPER** and **IF WTOR** on expiration processing. For both the **IF WTOR** and the **ASKOPER** script command has been changed so that, if the expiration timer is set via the **@9999h|m|s** script command setting or the **MAXWAIT=9999h|m|s parameter** (or via it's script setting counterpart) that instead of expirations ALWAYS defaulting to the second (option "B") setting, you can enclose the two possible answers "A" or "B" within parentheses so that the enclose option is the default. If both are enclosed or if both are NOT enclosed, then "B" is still the overall default on a timer expiration. When one of the above script commands expires instead of being replied to, the **IF EXPIRED** setting is set "ON" or "YES" or "TRUE". So that the commands between the **IF EXPIRED** and the next **ELSE** will be executed. Otherwise, if the timer has **NOT** expired, then the script commands between the first **ELSE** after

		the **IF EXPIRED** command and the **ENDIF** will be executed instead. Added **MSGDROP** to cause console processing logic to drop the message currently in use and move to the next match. This is only necessary on occasions where there may be multiple matches in the console message buffers and you want to move to the next one. It was discovered that (sometimes) when processing a countdown timer field, that using "@" for the countdown timer designator can cause issues with not only looking for messages that have an at-sign in them, but also with some of our other commands (like IFWORD, IFSTRING and MSGSTRING) which depend on that at-sign to show the start column or where to start within the generic word. Therefore, we have changed the countdown designator to left and right parentheses "(22s)". This actually make it easier to use in that (for instance **IF WTOR** and **ASKOPER**), you can already designate which of the two options should be used by default. i.e: **ASKOPER (2m) (happy),sad,How do you feel today?** The above script command would set a countdown timer of 22 minutes and the default if that timer should expire is the (happy) selection, which is option 'a' of the command with "sad" being option 'b' of the command. Normally "sad" is the default if no other default is set. Added ability to modify any script task that is running, (and waiting for anything) to use HOLD, PAUSE, WAIT or FREEZE to cause the task to enter a hold state. Entering KILL, or STOP will cause it to end (immediately). Wait FOR VTAM has been changed to WAIT UNTIL VTAM ACTIVE Added STICKY and UNSTICK commands STICKY is just like WTOH except that when the script ends, the message sticks to the console (until it is cleared, or it rolls off in console roll mode). UNSTICK is used to remove the STICKY note placed on the console with STICKY. It can only be used to remove the most current highlighted note, and only applies to STICKY notes added with the current script. (in case you want to issue "other" notes, and get rid of the previous one before you do).
9.0.1	11/11/2020	Added the ability to NOT recheck the stopped condition of a job after a long wait. For instance, if you specify: **WAIT UNITL STOPPED CICS22 30s** Syzcmd/z will wait for CICS22 to stop and then wait another 30 seconds and then check again that it is still down. It appears that there may be times when the site wants to wait the extended time after the task stops, but they do not want to be

		bound by the fact that it has to "still" be down at the end of that wait. So you could instead specify: **WAIT UNTIL STOPPED CICS22 30s NORecheck** Which will cause the same wait for CICS22 to stop and then wait another 30 seconds before continuing with the script, but if CICS22 is restarted before the expiration of the 30second wait, Syzcmd/z will NOT recheck to see if the task is restarted. Fixes to the started and stopped wait times to base the wait on multiples of the check interval instead of a static 1 second check. Therefore, if you have the interval set to 5s in the startup parms or in the script, Syzcmd/z will wait 5 seconds between scans to see if the task has started or stopped. The overhead of checking every second is fairly trivial, but on VERY large systems with thousands of addressspaces active, it is conceivable that it might make some sort of overhead difference, (but we doubt it). Enhance **IF EXPIRED** to allow it to be set for the rest of the script (or the next IF with a expiration possibility), so that the script is not "forced" to have the **IF EXPIRED** immediately after a "IF" which "may" expire.
9.0	05/19/2020	New Version – several new command processing modules Altered ESTAEX format to catch errors IF timers -- when asking operator questions (ASKOPER) or waiting for things to be responded to (IF WTOR) or "happen" (other IF's), a countdown timer can be set (via MAXWAIT or via individual options on the script commands themselves) which will automatically respond with whatever the OPTION "B" would be for that IF command. For instance, if you were to ask a YES or NO question, and you asked it such that YES was the "A" response and NO was the "B" response, then if the timer expires the "B" response (NO) would be used: For example: IF WTOR @45s **YES**,**NO**,Do you want to IPL &SYSID? Then if the operator has not responded to the question with a "YES' or "NO" within 45 seconds, then "**NO**" would be taken as the default response because it was the OPTION "B" of the request. If you were to specify the request as: IF WTOR @45 **NO**,**YES**,Do you want to IPL &SYSID? Then "**YES**" would be the default after 45 seconds, because it is located in the "B" option position. Position "A" and "B" can be up to 23 characters and can contain embedded blanks, not quotation marks are needed. The Countdown part can be delayed by responding "WAIT" or HOLD" to the request instead of the OPTION "A" or "B" variables. When a countdown timer has expired, it sets the EXPIRED variable so that you can test via the new "**IF EXPIRED**" request. **IF EXPIRED** Do something if the previous request ended due to expiration **ELSE**

		The previous request did not expire, so do something else **ENDIF** Timer controls have been added for the PAUSE command. The countdown can be paused via the operator MODIFY command "F", via "**F taskname,HOLD**" ("wait" is the same as "HOLD". This will stop the countdown and when you respond "GO" the countdown will restart at the beginning of the countdown. For instance, if you are in a 30 second countdown and 15 seconds into it you enter "**F Taskname,WAIT**", then the script will prompt you with a WTOR asking you to reply "GO" when you are ready to continue. When you reply "GO", depending on the type of pause you were doing, it will wither continue with the pause as originally set, or (for some pause options), the pause will be restarted from the beginning. Please see the individual commands and options to see the actual settings for each. Variables for individual words of messages that have been trapped via some of the commands can now be set as variables within other commands or interrogated as those variables. Additionally, you can address them as &Wnn or &Wn. Thus &W01 and &W1 are the same variable. When processing messages via the virtual console interface, you can decide to parse the individual words of the message (or not). See the individual commands for the syntax. When IF nests are built, you can now skip individual ifs and it will skip the entire IF/ELSE/ENDIF for that particular IF. (see the individual commands for more information and the syntax) The DELAY command was long ago depreciated, it has now been completely replaced by PAUSE, and DELAY is now a alias of PAUSE. The WAIT for OMVS or WAIT for USS commands have been replaced with the standard **WAIT UNTIL STARTED** OMVS command, full OMVS capability has been added to the "WAIT UNTIL STARTED " command. Pause (for an amount of time in seconds (default) minutes or hours) is now interruptible and restart-able. At any time for any script in a pause(d) state, you can issue the **F taskname,WAIT** (or HOLD or PAUSE) and the task will halt and issue a request for you to tell it when to begin again. Should you reply "GO" or "CONTINUE" to that message, the pause will take up where it left off, meaning that if you were "paused" to 90 seconds, and you told it to continue (or GO) at 60 seconds after the message was first issued, it will start up with 30 seconds remaining. If the original wait time has already transpired, (i.e. it's 91+ seconds later from the original PAUSE), then the timer will be expired as soon as you enter the "GO" or "CONTINUE". If however, you

		were to issue "RESTART" to that message, it will go back and restart at 90 seconds again. You can also (when in a paused state), issue a "**F taskname,RESTART**" command and the current PAUSE will be restarted at the beginning of that PAUSE. SO if you are 75 seconds into a 90s pause, and you want it to go back to 90seconds, you can issue "**F TASKNAME,RESTART**" and it will wind back the clock for the pause you are on. In most cases, when a script is "waiting" for something to happen, it is in an interruptible state, so you can modify the task to STOP waiting (and end the script), or SKIP the current waiting and if within an IF clause, the entire clause is skipped (both true and false), or **HOLD** or **WAIT** or **PAUSE**, which causes the script to freeze and issue a operator request that you can respond to when you want to un-freeze the script. Wait until STOPPED now has a built-in 1 second "sanity check' delay after which it will recheck to see that the task is still "stopped". This became necessary on sites that may have multiple tasks with the same name (especially queued batch jobs) that could have one finish and another one start between the check interval of SyzCMD/z. It was found that a JOB named MYJOB could end, SyzCMD/z sees that it is now stopped, and because SyzCMD/z very quickly sees the task end, the script continues, even though another job may start immediately after (1/10 1/100 or even 1/1000th of a second) with that same name. This is especially troublesome on very fast systems and multi-processing systems where many tasks can be in operation and the signaling paths to the z/OS MASTER AddressSpace can get delayed. The default 1 second should be more than adequate. The 1 second delay is a default and can be changed to "NONE" or "OFF" via **STOPCHECK=NONE|OFF** or can be changed to any number of seconds minutes or even hours. Please be careful because the script will wait for the job to end, then pause for however long was set in **STOPCHECK=**9999h|m|s And then recheck to see that the task is still "gone". If it's not gone (i.e. another one with that name started), the script will again wait until it ends, then pause for the **STOPCHECK=**??? Time and try again. Waiting excessive amount of time is not very efficient, and while there are times when it may be useful, it's unlikely that anything more than a few seconds will realistically be necessary. So waiting via a different type of command (pause) is probably more useful since it can be halted or lengthen on the fly. Large processor sites which used to code two back to back "**WAIT UNTIL STOPPED taskname**" for the same task will no longer need to do that. Also, the "WAIT UNTIL STOPPED taskname" now has an additional parameter possible, where the site can denote how long they want SyzCMD/z to wait after that task is ended before the scipt continues. "**WAIT until stopped taskname 5s**" is the same as specifying **STOPCHECK=5s**, except that the "extra" wait would

		apply to just that one "**WAIT UNTIL STOPPED**" and not all of them. If there are some tasks which are prone to the "multi-tasks with the same name" effect, then they can be handled separately, rather than waiting for all tasks to end for the same amount of time. When the "**WAIT UNTIL STOPPED taskname 9999s\m|h**" option is used, the system parameter **STOPCHECK=NONE** can safely be set in the SyzCMD/s parmlib member. **Wait until ONLINE** or **OFFLINE** command has been enhanced to support both MAXWAIT and individual countdown timers on the command, similar to Wait until STARTED|STOPPED and several other WIAT related commands.
8.8	12/23/2019	Major change to IF DOW to allow multiple days and also changed the operand that allows the week that the day falls into to be specified, (i.e. -first SUNday, -second Tuesday , etc.) Added most of the known world-wide SMS text addresses for most (but probably not all) Cell providers. Added support for checking that SyzEMAIL/z is active before any email that might require special processing (i.e. that uses email groups, etc.) to be created.
8.7	2/19/2018	Added ability to check if tape drives are online or offline as well as the ability to check what VOLSER is mounted on the tape drive(s) Added the ability to pause the execution of a script via an operator command "F TASKNAME,HOLD" or "F TASKNAME,PAUSE", where TASKNAME is the name of the JOB or the TASKID of the STC that is running the script. Once HELD, the task will issue a WTOR asking to reply "GO" when ready to continue. The task will remain paused until either "GO", "STOP" or "KILL" is received. Go continues the task where it left off, STOP will end the task when the current command completes (but will cause a CC=12). "KILL" will end the task before the next command script command is accepted, and will cause the task to end with CC=99. Added the ability to stop the script task at any time via "F TASKNAME,STOP". This will allow the current command to complete and will stop the task before the next command is read from the script. The task will receive a CC=12 step code. Added the ability to KILL the script at any time via "F TASKNAME,KILL". This will immediately KILL the script before the next script command is executed. The task will receive a CC=99. If a task is in "HOLD" status, (i.e. the operator issued a "F TASKNAME,HOLD", you can "stack" other commands into the buffer. Currently only "HOLD", "KILL" and "STOP" may be stacked.

		Added LOOPCHK and LOOPCHEKC command. This command turns of the default GOTO loop checking within a script. This frees the site from the necessity to use "GOTO ++RESET" when creating a loop within the script on purpose. Added new parameter to the PAUSE command of "LONG or FOREVER". Either sub-parameter will issue a WTOR to the console and PAUSE the script until the WTOR is replied to. Replies can be "GO" to continue, or "STOP" or "QUIT" to end the script. To skip to the "next day" in a script, you can now PAUSE UNTIL 24:00, which will (just) AFTER midnight local time. You can do this any number of times to skip a day each time. Fixed a problem where if the site didn't include a //INTRDR DD card that a SUBMIT command would fail and get a S0c1 because the INTRDR was not open. The new code will dynamically add the DD if not provided in the task's JCL
8.6.1	08/15/2017	Add ability to dynamically change the STC TASKID (STCID) to make it easier to deal with multiple SyzCMD/z tasks running at the same time. It can also be displayed via &STCID Added &SCRDSN which shows the dataset that the script was being executed from (or inline of DD *) Added &SCRALL which shows the dataset and member name of the script, (or INLINE if DD *) Fixed &SCRIPT to show the member name (or INLINE if DD *) Added INTerruptible option to PAUSE command to allow all pauses (both UNTIL and for an amount of time) to be SKIPped or to allow the script to be STOPped during the pause. All variables can be used in IF statements. i.e. instead of IFTASKNAME = CICS001 you could use IF &TASKANME = CICS001 Any variable (valid variables start with ampersand, "&"), can be used in this way. The email related TO: and FROM: commands have been changed so that they can be used both within and outside of the EMAIL portion of a script. You can compare and set them before (or during) the email creation. Previously, they could not be used unless you actually had started to generate the email in the script. BEFORE and AFTER no longer require leading zero for times before 10am, i.e. BEFORE 09:00 can be used as BEFORE 9:00. TASKTYPE is now an alias of MYTYPE or TYPE to make using the command easier (to understand).

		When a log file (files that can be used to write data to during script execution) is opened and closed, the dataset name is displayed as well as the number of lines/records added to the dataset. It was noticed that SyzCMD/z sometimes executes so fast that the times of the consequences of the commands can be before the command appears to have been issued. Timings will no longer be rounded for some items, and the actual time will be kept (internally) to 4 decimals HH:MM:SS.nnnn If a script ends without reaching an ENDIF command, the return code is set to "4" to denote a low impact script programming error.
8.6	12/17/2016	Version 8.6 adds support for console message monitoring. This feature is not meant to replace the much more comprehensive console message monitoring features of SyzMPF/z, but it does allow the SyzCMD/z scripts the ability to interactively process all of the console messages generated while the script is running. New Internal Commands in 8.6: • MSGSTR – allows scanning of all console/syslog messages to find a specific (case sensitive) "string" within that message. There are controls to scan the ENTIRE message or simply beginning in a specific column of the console/syslog message. At the completion of this command the first 50 words of the message found are available as variables within the script and can be used at any time. • MSGWORDS – allows scanning of all console/syslog messages to find a specific set of words (up to 6 case sensitive words) within the console/syslog message. The words may appear in any order, but all words identified MUST appear somewhere in the message. At the completion of this command the first 50 words of the message found are available as variables within the script and can be used at any time. • MSGWORD – same as MSGWORDS except that it looks for a specific Message text (case sensitive) word in a specific place within the message, i.e. a message that the third word is 'FRED'. At the completion of this command the first 50 words of the message found are available as variables within the script and can be used at any time. The same word position can be checked for multiple alternate words (up to 6). • IFSTR – (If String) Works the same as MSGSTR (above), but allows the use of the IF/THEN/ELSE logic of the SyzCMD/z scripts. At the completion of this command the first 50 words of the message found are available as variables within the script and can be used at any time.

		• IFWORDS – (If WORDS) works the same as MSGWORDS (above), but allows the use of IF/THEN/ELSE logic of the SyzCMD/z scripts. At the completion of this command the first 50 words of the message found are available as variables within the script and can be used at any time. • IFWORD – If individual WORD) works the same as MSGWORD (above), but allows the use of IF/THEN/ELSE logic of the SyzCMD/z scripts. At the completion of this command the first 50 words of the message found are available as variables within the script and can be used at any time. The same word position can be checked for multiple alternate words (up to 6). • ACTCONS – Activates an internal console of the specified name (as a parm of this command), so that SyzCMD/z can monitor the generated messages for use by the above new commands. • DEACTIVATE CONSOLE – deactivates the console that was activated via the (above) ACTCONS command. • &W0-49 – variables that are filled with the first 50 words of the console message located via one of the new console message processing commands. Words are numbered relative to zero (being the first word). &W0 is normally the message number for “normal” messages, operator commands that are scanned will not normally have message numbers so &W0 will be the first word they typed at the console. • &WORD variable is set to the word in the Message that matched the requested position within the message and is set by the “IF WORD” and “MSG WORD” command requests. • &STR and &STRING variables are set to the string that matched the requested position within the message processed and is set by the “IF STR” IF STRING” “MSG STR” and “MSG STRING” command requests. • The PAUSE command has been enhanced to allow the generated information message on the task waiting to be “other” then the default HIGHLIGHTED. It can be Low or not displayed at all with the new feature. • LOG command which allows you to dynamically write to a “log” record in any existing (or new) dataset (or SYSOUT) so long as it adheres to the size constraints of the product. Optionally specify allocation parameters of CYL|TRK as well as primary and optionally secondary space for the new dataset. The dataset may be closed and reopened as many times as may be desired. • LOGDSN command and startup parameter to give the ability to dynamically change the LOG dataset as many times as you want during execution as well as have a “default” LOG dataset for all scripts. The SyzCMD/z tasks now display the total CPU (TCB and SRB) time used at the end of the execution of each task. Also displays the total run time (wall-clock).

		SyzCMD/z tasks will now display the task name in all informational messages so that when multiple tasks are waiting for "something" you can tell which one is which. Pause UNTIL now allows H:MM instead of always HH:MM so you can specify Pause until 1:23 instead of always 01:23.
8.5	5/19/2016	Version 8.5 adds better cross capability with SyzMPF/z so that they share a common command base. Both products now have the same basic commands. New Internal Commands in 8.5 • IF (anything) – you can now create an IF based on any value, variable or name. All internal and external variables are now supported and can be compared (even generically) to almost anything. • REPLY command can retry up to 99 times when a message is not matched (yet). The interval for the retry is controlled by the standard "INTERVAL=nnnn|s|m|h" command. • IF WTOR – now you can ask any question and obtain the response from the operator and act IF/THEN on it. • IF RESPONSE to be able to perform logic on response received from ASKOPER or WTOR command(s). • &RESPONSE – this is the response sent back from the IF WTOR and ASKOPER commands • IF PGMRname – to check the programmer name data on the task. • IF ACTive is now an alias of IF STARTed • IF INACTive is now an alias of IF STOPed • All IF's (where appropriate) can now compare > (greater), < (less), GE (greater or equal), LE (Less or Equal), NE (Not Equal), = (Equal), etc. instead of just equal and not equal. • IF STARTed/STOPped/ACTive/INACTive can check on jobs generically. Instead of just CICS1234 it can check CICS* and any CICS will suffice. Generics are represented by "%" for single character and "*" for all characters. • Full EMAIL support of MAXCC (some parameters will require the SyzMAIL/z product to operate). • Email attachment support for all JES datasets (input JCL, JCL, JESMSGLOG, JESYSMSGs, etc.) Any single JES dataset, combination or all of them can be attached. • Email attachment support of any Sequential file or PDS member. (current 8.5 limit is 3 dataset attachments) • ASKOPER command to ask anything of the operator and get back the response in &RESPONSE. • DOMALL command. This will allow SyzCMD/z to remove ALL highlighted messages (both WTOH and errors) that it has produced during the execution. • SUBMIT – submit any job from a sequential dataset or PDS. • REXX Return Code now saved from execution of EXECs.

		• IF MYSUBSYStem control command has been added to allow testing if the task is operating under the Master Scheduler (MSTR), JES2, JES3 APPC, OMVS, etc. • IF SUBMETHod control command to allow script execution based on the way this script entered the system, INTRDR, READER, REMOTE, etc.
8.4	7/10/2015	Version 8.4 fixed some minor message formatting issues when more variables than will fit on the scaled line are used. Also there was a problem on some (z/OS 2.2 test) sites when static and persistent variables are swapped which was fixed by setting a fixed swap length.
8.3	1/23/2015	Version 8.3 fixes and adds the following features: • the RACROUTE command errors for GLOBAL=YES under z/OS 1.12 and higher (especially z/OS 2.1) • SyzMAIL=YES now forced if task is executing SUB=MSTR because no JES areas are available. • Corrected minor error on return from Master Scheduler table starting on z/OS 2.2 • Added trace capability for Email creation
8.2	7/11/2014	Version 8.2 contains minor fixes to the static variable content and support
8.1	1/23/2014	Version 8.1 adds the following Features: • Email support – uses new SyzMAIL/z product To:, From:, Cc:, Bcc:, Subject:, ReplyTo: Message support • Execution Class checking (&EXECCLASS & IF EXECCLASS) • MSGCLASS checking (&MSGCLASS and IFMSGCLASS) • Step and/or procstep result checking • Workload name checking (&WORKLOAD and IFWORKLOAD) • RACF group checking (&RACFGRP and IFRACFGRP) • NOTIFY ID checking (&NOTIFY and IFNOTIFY) • Submit method checking (&SUBMETHOD and IFSUBMETHOD) • Additional Installation Defaults supported via Parmlib member • Generic characters (‘%’ and ‘*’) are now permitted for MSGTASK, TASKNAME, MSGID, LPAR, SYSID, and SYSPLEX • Debugging mode • Greater variable support &Maxcc, &Stepcc, &MaxSTEP name, &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 variable of eMail • Variable support for SyzMail/z
8.0	8/23/2013	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. This is a control command which allows the site to turn off variable processing. By default, SyzCMD/z will parse each script line to see if it contains variables that it may need to populate. This overhead is extremely minor, but if you are absolutely certain that you will not be using variables in your script, you may use the VARIABLES control command to turn off the check of each script line to see if a variable exists. New Support for NON-persistent variables. The support for NON-persistent variables is very similar to the persistent variable support added in Version 8.0 except that when the SyzCMD/z task ends, the NON-persistent variables are automatically removed from the system. This allows a great deal of logical support. Support for deleting persistent and NON-persistent variables has been "enhanced" to make the process easier. Support added to test the success of the variable sub-command. The &LVRC (Last Variable Return Code) variable can be tested and displayed with new script commands. &LVRC display variable and insert variable added. IF LVRC script command added to allow for testing of the persistent and non-persistent variable commands. 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. To **set** a new variable: Persistent variables SETVAR <name> variable-contents SETVARP <name> variable-contents Non-Persistent (temporary) variables SETTVAR <name> variable-contents SETVART <name> variable-contents SETVARNP <name> variable-contents To **Replace** a variable: Persistent variables

REPVAR <name> new-variable-contents
REPVARP <name> new-variable-contents

<u>NON-Persistent variables</u>
REPTVAR <name> new-variable-contents
REPVART <name> new-variable-contents
REPVARNP <name> new-variable-contents

To **<u>Delete</u>** a variable:

All forms of the delete are equivalent, DELETE does not consider any difference between persistent or non-persistent variables. The various spelling formats of the DEL command are supported just for compatibility.

DELVAR <name>
DELPVAR <name>
DELTVAR <name>
DELVART <name>
DELVARNP <name>

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)):

&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
&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 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. New ESTAEX support was added to handle any ABEND the may happen even if it is outside the SyzCMD/z code. ASIS command added so that any script line that the site wants to contain an ampersand “&” will not be treated like a variable and replaced (or attempt to replace) the &word with a set variable. EMAIL support: The site can now send eMail from within a SyzCMD/z script. Full support for FROM:, TO:, REPLY-To:, CC: (up to 5), BCC (up to 3), Subject:, and any length message (up to the max SMTP supported length) can be sent. The message can contain variables that are replaced at eMail send time. A default From: and To: can be set in the system parmlib concatenation member for SyzCMDz (SYZCMD00), and a new TimeZone parmlib setting is also available. The site can send any number of emails from within a script. It is planned to allow IF/THEN support within the email message itself, but currently the IF/THEN/ELSE support is only supported for all parts of the email EXCEPT the actual email text (body of the email). A system REXX command “BPXSTOP” is available upon request which can be used at shutdown of z/OS and it will stop all USS/OMVS processes that are active. This can be performed before the “F OMVS,SHUTDOWN” command to get rid of all threads and processes that are “difficult” for OMVS to end on it’s own. SyzCMD/z will DOM any highlighted messages that were created during execution at end. This can be turned off via a parm or command during execution of SyzCMD/z. SyzCMD/z now fully supports checking for the existence of a dataset, whether it is cataloged, or actually exists, or even the create date, last reference date (date last touched) or if the dataset exists on DASD (or tape) or has been migrated. Full IF/THEN/ELSE support is provided by the IFDSN command. Future

		support should allow for testing the dataset variables (organization, LRECL, Blocksize, dataset size, etc.) Support for a Maximum WAIT time now exists, in the format of: MAXwait=nnnnH/M/S The MAXWAIT parameter allows the site to set a maximum amount of time that ALL wait related commands can wait for an event to happen. The default is "forever".
7.3	5/23/2013	• Added MLWTO generation support. You can now generate a Multi-Line WTO instead of a bunch of separate lines which might become separated. The new MLWTO= command supports up to 99 lines in one message. • SyzCMD/z now displays the library and script name that is being processed. • new totals fields added.
7.2	7/22/2012	Minor updates and fixes, product now fully supports SYZFMT messaging.
7.1	3/23/2012	• Added IF MYOWNER command to allow the site to test the USERID of the person or task that submitted the SyzCMD/z task • Added IF MYSTEP command to allow the site to test the STEP name of the task (JOB or STC) that SyzCMD/z is currently executing. • Added IF MYPSTEP command to allow the site to test the PROC STEP name of the task (JOB or STC) that SyzCMD/z is currently executing. • Added IF SUBMSTR command to allow the site to test (Yes or No) of whether SyzCMD/z is currently executing under control of the Master Scheduler or not (i.e. under JES2 or JES3) • Added IF TYPE command to allow the site to test if the SyzCMD/z task is running under control of a Batch job (JOB), Started Task (STC), TSO User (TSU) or APPC (APP). • Changed SyzCMD/z to allocate the internal SYSTSPRT dataset (for REXX commands) to DD DUMMY if we are executing under the Master Scheduler (and not JES) and the site has not specified a existing dataset or DD to use. You cannot use a SYSOUT dataset if operating under the Master Scheduler, only if under JES2 or JES3. • Added much more descriptive failure messages for Dynamic Allocation Failures so that they can be immediately corrected at the site.

		• Corrected several commands that were not generically handling using an '=' in place of a space between the command and the parameter operand. • Remove restriction of number of spaces between command and the parameter operand to make scripts more "free form".
7.0	9/28/2011	• Added the ability to "GOTO" labels within the command script with the new "GOTO" command. This capability adds the ability to set labels (::labelname) to give you anchors for the "GOTO" command. Logic is set up in SyzCMD/z to detect a possible loop condition. If a loop to the same ::labelname is detected twice in a row, it is assumed to be a "LOOP" and will cause the script to terminate. You can override this behavior by using the "GOTO=++RESET" option somewhere after the ::labelname. The "++RESET" will "reset" the loop counters and allow you to branch to that label an unlimited number of times. Code is being worked on to provide for "DO" and "LOOP" commands, which will remove the necessity for the ++RESET. Until that time, to set up a "LOOP" on purpose, you must use the ++RESET option to keep from detecting an invalid loop condition. • Added the SHOWSYSTEMVALUES command to show z/OS system, LPAR and physical hardware values on the system console and syslog. This command can be used on any LPAR to show the values which can be tested with various other IF-logic commands of SyzCMD/z. • Added the REXX command which allows execution of REXX exec from within SyzCMD/z. • Added IFREXCC which provides the ability to interrogate the return code from the REXX command issued through SyzCMD/z and to take IF/THEN/ELSE actions based on the return code. The operations allowed with the command are "=", ">", and "<" (Equals, Greater than and less than). • The system parmlib can now contain a SYZCMDnn startup member, which can control the base state of the SyzCMD/z program. • The Commands DD is no longer required for operation. The Commands dataset can instead be defined in the System Parmlib member. In fact, no DD's are required for SyzCMD/z operation, all of them are able to be dynamically controlled during operation of SyzCMD/z. • Added the SIMULATE script command to allow the site to Simulate the commands to be issued or Replies to be responded to.

		• Added the GOBACK command to provide a method of getting back to the base script from a ++INCLUDE (sub-script) quickly. • Several system defaults can be set via the SYZCMDnn startup member of the system parmlib concatenation, including the command library, whether or not the WTO's are displayed by SyzCMD/z (license information) on the console, ECHO defaults and others. • Added ++INCLUDE script command to allow the site to include other script members of the same (or any via the INCDSN= subcommand or setting) library to be processed as if included in the base script member. • Many of the WAIT based commands can now be interrupted before they complete. Several options are available, including the ability to treat the wait as if it was successfully satisfied or to stop waiting and end the script. The affected commands have had their informational WTO's changed to add the text "Task modify available". • Merge EXIT and STOPCODE logic so that EXIT will be able to set condition codes like STOPCODE.
6.1	1/23/2011	Version 6.1 • Added new INTERVAL command. Prior to this control command, SyzCMD/z would always wait 15 seconds for time related events to be performed. For instance if SyzCMD/z were waiting for a task to start (or stop), it would check, sleep for 15 seconds then try again. If the task that SyzCMD/z was waiting for started even 1 second after the timer was set, it wouldn't check again for 15 seconds. The overhead of testing more frequently is infinitesimally small, so it was decided (based on customer requests) to allow the 15 second value to be changed. The INTERVAL command allows you change that default 15 second value. 15 seconds is still the default. • This manual adjusted to place all control commands in alphabetical sequence to make them easier to find.

Table of Contents

Contents

1. Introduction

SyzCMD/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 site to schedule any system or subsystem command, submit batch JOBs and control timed operations on a 24x7 basis.
>
> SyzCMD/z, the Command Scripting facility, which allows the site 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 site to respond to any Console-type message with predefined scripts allowing complete interactive control of any console situation or message. SyzMPF/z allows 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, provides a mechanism for all Syzygy programs to interactively send eMail as well as providing the end of task notification facility via eMail.

Besides sending general eMail messages, SyzMAIL/z captures the maximum condition code, plus all (or any) of the Step condition codes, plus other task related information 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.

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.

Related Documents

SyzAUTO/z	Installation and Operations Guide
SyzCMD/z	Installation and Operations Guide (this guide)
SyzSPOOL/z	Installation and Operations Guide
SyzMPF/z	Installation and Operations Guide
SyzMAIL/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 SyzCMD/z utility is designed to provide the “authorized” user with the capability of responding to system messages and/or executing simple or complex “scripts” of MVS, JES2, JES3, CICS, DB/2, or Unix System Services (USS) commands from within a batch job or started task. The command(s) can be issued from the PARM parameter on the “EXEC PGM=” card or from the parameter script file both.

SyzCMD/z can be used to automate the operating system IPL, Shutdown, and almost any operator function in between. The product can be called from within “authorized” high level programs to execute commands and perform functions that would otherwise not be possible.

SyzCMD/z is not simply a “dumb” scripting facility for issuing commands, it has a direct interface with your site’s security system (RACF, ACF/2, TOP/Secret) and will allow an “authorized” user to use the many special features for determining what needs to be done and when. The utility provides a very simple scripting language that allows the “authorized” user to harness extremely powerful command logic, including nested IF/THEN/ELSE logic.

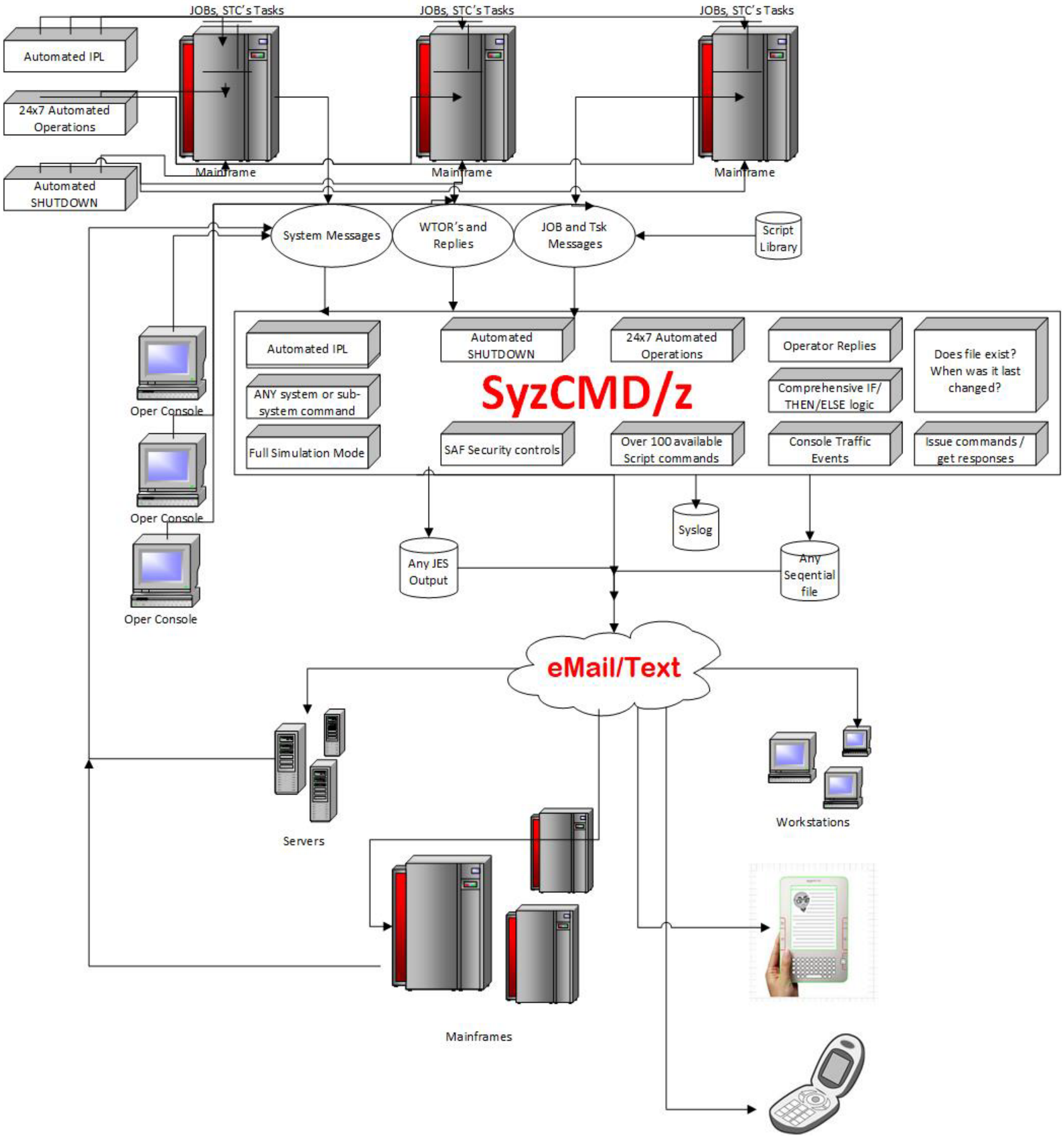
Facilities are provided to allow the “authorized” user to perform script/command logic only IF some task is active, or not active, or only if (or if not) a certain day of the week, or time of day.

There are over 35 “special” internal logic commands built in to the SyzCMD/z facility with more under development.

SyzCMD/z also provides the ability to respond to console message traffic with any of the scripting commands, whether the message is a WTO or a WTOR, even if the message ID contains up to the maximum 17 characters. SyzCMD/z will not only know what message is being processed, but you are able to interrogate for the name of the task that issued the message. So if a CICS region issues a message, you will not only know the message, and be able to act on it, but you will know which CICS region issued it.

Functional Description

SyzCMD/z is meant to be run as a JOB or TASK step, or as a console started task. It is possible to run as a called program from other programs (i.e. from within a COBOL program), but the main intention of SyzCMD/z is to run as a separate task or step. SyzCMD/z is composed of several 24 bit, 31bit and 64bit subroutines (CSECTS) generated into a single load module, which must be executed from a “APF authorized" library. Further, many of the functions of SyzCMD/z are protected by calls to the sites Security system (RACF, ACF/2 Top Secret, etc.), and care should be taken to provide necessary security system authorization to users, subsystems and tasks which are meant to execute SyzCMD/z



3. Installation instructions

Procedures

STEP 1:

The software you have ordered is 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 directory. This will create a binary file (in TSO TRANSMIT format) with the following name structure: software.XMI 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 ON YOUR MAINFRAME

Pre-allocate a file on MVS (e.g. 'userid.software.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 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" file 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 CNTL PDS

On your MAINFRAME, either from a TSO/ISPF session or a TSO batch TMP (i.e. EXEC PGM=IKJEFT01), execute the TSO RECEIVE command on the uploaded XMIT file from step 4 as follows:
(*note If operating under ISPF use panel option 6 (“Enter TSO or Workstation commands”) or prefix the RECEIVE command with “TSO” to have it directly execute by TSO instead of being interpreted as an ISPF command, or exit ISPF to TSO READY state.)

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

***where 'userid.software.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 should enter:

```
DA('userid.software.loadlib')
```

... 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 dataset.

PLEASE BE SURE TO CHOOSE A 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.loadlib')  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).

STEP6:

***REPLACE THE 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. 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 (optional):

Changes required to process console message traffic. This capability is better suited to the SyzMPF/z product, but SyzCMD/z does possess limited console message processing capability.

To process console message traffic, you will use the SYZMPF MPF exit program. This program was delivered with your base SyzCMD/Z program and should be copied to a authorized load library in LinkList.

In order to activate SyzCMD/z for a message you need to tell the system what message(s) you wish SyzCMD/z to process. It's triggered by placing an entry in the active system parmlib MPF list, and a corresponding member of the Command Scripting dataset is necessary. 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 MPF list member is (MPFLST00) :
Contents of "SYS1.PARMLIB(MPFLST00)

```
.....
*
*       DFHSI1517,SUP(NO),USEREXIT(SYZMPF)
*
*       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.COMMANDS(DFHSI151)" which will contain
*       The script logic and possible commands that are to be issued
*       in response to DFHSI1517.
```

Contents of “SYS1.COMMANDS(DFHSI151)”

```
*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:
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
```

STEP8 (Highly Suggested):

Add the SYZCMDnn parameter to your System Parmlib Concatenation.

In order to support greater control over the capabilities of SyzCMD/z, a new feature was added to version 7.0 of the product which allows the site to control overall operation of the SyzCMD/z product. This member, (which can exist in ANY level of the SYSTEM PARMLIB concatenation), will hold all of the “defaults” for global SyzCMD/z operational capabilities. If this member does not exist, then SyzCMD/z operates the same as pre-version 7 facilities.

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

The default member name is SYZCMD00 (zero-zero). You can override that default at run time, (PARM=’ID=xx), 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, thus to have comments in the member, you “should” use a “*” (asterisk) in column 1 of the line, and while it is not absolutely necessary, it would be a good idea because at some future time the restrictions on “invalid” parameters might be implemented.

Sample JCL and Possible RACF Changes.

Normally you should add a proc to your system proclib so that you can easily use the facility. Most sites add 3 procs (AUTOIPL, SHUTDOWN and COMMANDS), they are identical, except for the contents of the override. This is so that you can authorize the AUTOIPL and SHUTDOWN procs to and allow them to run without any operator intervention. The less people have to remember to type the better, thus, when it's time to shut the system down for IPL, they can just type:

S SHUTDOWN,SUB=MSTR (the SUB=MSTR allows the command scripts to control the complete shutdown of the system including JES2, without it, the command script runs under the control of JES2 and thus can't be used effectively to control anything once JES2 is attempted to be stopped).

The sample JCL is as follows:

Sample AUTOIPL JCL

```
//AUTOIPL  PROC M=AUTOIPL
//*
//*       OPERATOR COMMAND PROGRAM
//*
//COMMAND EXEC PGM=SYZCMDZ,TIME=1440,PARM='ID=IP'
//*                         use SYZCMDIP parmlib member
//IEFRDER  DD DISP=SHR,DSN=SYS1.COMMANDS(&M)
```

*Note. The SYS1.COMMANDS dataset is a PDS that is allocated as Fixed Block (FB) with an LRECL of 80, any blocksize is acceptable, but ½ track is best, this dataset will be used to contain all of the scripts for Command Processing.

Sample SHUTDOWN JCL

```
//SHUTDOWN PROC M=SHUTDOWN
//*
//*        OPERATOR COMMAND PROGRAM
//*
//COMMAND EXEC PGM=SYZCMDZ,TIME=1440,PARM='ID=SH'
//*                          use SYZCMDSH parmlib member
//IEFRDER  DD DISP=SHR,DSN=SYS1.COMMANDS(&M)
```

Sample "Normal Command" JCL

(default name is COMMANDZ)

```
//COMMANDZ PROC M=DUMMY,ID=00
//*
//*        OPERATOR COMMAND PROGRAM  ID=00    = (SYZCMD00) parmlib
(default)
//*
//COMMAND EXEC PGM=SYZCMDZ,TIME=1440,PARM='ID=&ID'
//IEFRDER  DD DISP=SHR,DSN=SYS1.COMMANDS(&M)
```

The Normal command JCL proc can be used as follows:

Assume the actual proc is called "SYS1.PROCLIB(CMD)" for example.

S CMD,M=TCPIPUP

Will point to the command script "SYS1.COMMANDS(TCPIPUP)" to bring up TCPIP.

*NOTE

If you decide to call your “normal” proclib JCL member something other than “COMMANDZ” AND you wish to have SyzCMD/z MPF processing available (instead of the SyzMPF/z product) you will need to contact Syzygy at (800) 767-2244 to obtain a zap job to alter the MPF processing program to use the alternative Proc member name

RACF changes that may be necessary for some sites:

Depending on your sites configuration, you may or may not have to perform RACF changes to create these procedures. If so, then the following will be necessary:

```
ADDUSER SHUTDOWN DFLTGRP(STCGROUP) NOPASSWORD NOOIDCARD

CONNECT SHUTDOWN GROUP(STCGROUP) AUTH(USE)

CONNECT SHUTDOWN GROUP(SYS1) AUTH(USE)

RDEFINE STARTED SHUTDOWN.* STDATA(USER(SHUTDOWN) +
  TRUSTED(YES) GROUP(STCGROUP))

SETROPTS RACLIST(STARTED) REFRESH
```

Also, each user that you wish to be able to use SyzCMD/z via a submitted batch JOB (not as started task, as we covered that above), will need to have access to the FACILTIY CLASS profile of SyzCMD/z. You may have to do this for all users who are not in the SYS1 group already:

To allow all users in the SYS1 group (only if necessary):

First define the class and don't let anyone use it.

```
RDEFINE FACILITY SYZCMDZ UACC(NONE)
```

Now let all the users in the SYS1 group get to it.

```
PERMIT SYZCMDZ CLASS(FACILITY) ID(SYS1) ACCESS(ALTER)
```

Anyone else that you wish to allow to use SyzCMD/z either internally (called from a program) or via a batch JOB will need to be permitted, but they should be permitted as READ not ALTER as follows:

```
PERMIT SYZCMDZ CLASS(FACILITY) ID(the user id) ACCESS(READ).
```

SyzCMD/z has the capability to limit access to certain commands based on the RACF ACCESS. This capability is controlled via a special service-pack update which is available upon request from Syzygy. This facility is low use, and increased system overhead and is therefore not suggested for standard operation.

4. Reference

PARMLIB Startup Member Parameters:

Sample SYZCMD00 “SYSTEM PARMLIB” DEFAULTS member

```
*
* This is the V8.x SYZCMD/z startup parm member
*
* CMDDSN=Data.Set.Name          DSN where the command scripts are kept
*
* CMDMEM=MemberName             Member of CMDDSN (or //IEFRDER) that
*                               will be used by default.
*                               (older OS's require a dummy member
*                               which contains no actual commands)
*
* Debug=yes|NO                  Show module name on Echo
*                               Default is NO
*
* ** * ** * ** * ** * ** * ** * ** * ** * ** * ** * ** * ** *
* EMAIL-RELATED Commands
* TO:Email Address              Default TO address
*
* CC:Email Address              Default CC address
*
* FROM:Email Address            Default FROM address
*
* MAXMSG=nnnn                   maximum text message allowed (dflt=50)
*
* REPLYTO:Email Address         Default Return address
*
* SUBJECT:anything              Default Subject
*
* TIMEZONE=+/-nnnn or USzone    Time zone of this system (defautl is PST)
*
* SyzMail=yes\No                SyzMail Product Support
* ** * ** * ** * ** * ** * ** * ** * ** * ** * ** * ** * ** *
*
* INCDSN=Data.Set.Name          DSN where commands scripts are to be
*                               INCLUDed from (defaults to CMDDSN
*                               or //IEFRDER)
* INCMEM=MemberName             Member of INCDSN (or //IEFRDER) that
*                               will be used.
*
* INTERVAL=nnnnH|M|S            Interval for wait and retries. max of
*                               9999 Hours, Minutes, or Seconds
*
* Maxwait=nnnnH/M/S/Forever     Sets maximum time that any wait command
*                               will be allowed to wait before the
*                               wait is ended, sets EXPIRED variable
*                               that can be checked with IFEXPIRED
*                                Default is:FOREVER
*
* RACF=Y|n                       Check RACF facility class to authorize
*                                Default is Yes
*
* SIMULATE=y|N                   Simulate all commands
*                                Default is No
*
* STATistics=all|verbose|DEFAULT|none  Show shutdown stats
*                                      ALL and Verbose show all stats
*                                      NONE shows no stats
*                                      Default shows important stats
*
* STARTUP=QUIET                  Don't show any of the startup HELLO msgs
*
* STEPLIB=Y|OK|n                 Allow SyzCMDz/s to execute if //STEPLIB
*                                exists in the step  Default=Yes
*
* SYSEXEC=Data.set.name          DSN where REXX execs are loaded from.
*                                or =Data.set.name(member)
*
* SYSOUT=Class                   Default SYSOUT class (default is A)
*
* Verbose=yes|NO                 Show long wait messages or short ones
*                                Default is NO (short ones)
*
CMDDSN=Sys1.Commands
*
INTERVAL=15S
*
INCDSN=Sys1.Commands
*
SYSEXEC=USER.Z110.CLIST
*
STEPLIB=OK
*
*STARTUP=QUIET
*
*CMDMEM=NOTHING
*INCMEM=NOTHING
*
RACF=NO
*
SYSOUT=F
*
TO:Products@SyzygyInc.com
FROM:SyzCMDz@SyzygyInc.com
```

```
CC:JOBlog@SyzygyInc.com
MAXMSG=100
*
TimeZone=-0700
*
Statistics=default
*
Syzmail=Yes
*
END of MEMBER
```

CMDDSN

data.set.name | data.set.name(member)

The CMDDSN parameter allows the site to specify a default dataset (or dataset and member, which is like performing CMDDSN and CMDMEM together) which will contain the command scripts. If the running task specifies a //IEFRDER DD, it will override this specification. However, using this parameter will allow you to eliminate the //IEFRDER DD completely from your task JCL.

NOTE: Tape data sets and migrated data sets are not supported.

Example:

```
........
* This is an example of SYZCMD00 Startup Parameter member
**************************************************
CMDDSN=SYS1.COMMANDS
**************************************************
INCDSN=SYS1.commands
SYSEXEC=USER.CLIST
STARTUP=QUIET
*CMDMEM=NOTHING
*INCMEM=NOTHING
RACF=YES
SYSOUT=F
SYZMAIL=Yes                    ←Allow email Nicknames to be generated
MAXMSG=100                     ←Default is 50
TZ=-0700                       ←Time zone is Pacific time (-700)
To: AllMail@SyzygyInc.com      ←Default to use if script forgets or is invalid
From: SyzCMDz@SyzygyInc.com    ←Authorized to send mail from this LPAR
CC: JOBLOG@SyzygyInc.com       ←send copy to JOBLOGs
*
```

CMDMEM

membername

The CMDMEM parameter allows the site to specify a default member (of the provided CMDDSN dataset) of one (1) to eight (8) characters which will contain the command script to be executed. This parameter can be specified in the EXECution parm of the SyzCMD/z task instead of pointing to the member in the //IEFRDER DD.

NOTE: Only one member is allowed in the parmlib defaults dataset, but any number can be used in the execution parms separated by “;”s.

Example:

```
........
* This is an example of SYZCMD00 Startup Parameter member
CMDDSN=SYS1.COMMANDS
INCDSN=SYS1.commands
SYSEXEC=USER.CLIST
STARTUP=QUIET
************************************************
CMDMEM=DEFAULT
************************************************
INCMEM=INCLUDE
RACF=YES
SYSOUT=F
SYZMAIL=Yes                      ←Allow email Nicknames to be generated
MAXMSG=100                       ←Default is 50
TZ=-0700                         ←Time zone is Pacific time (-700)
To: AllMail@SyzygyInc.com        ←Default to use if script forgets or is invalid
From: SyzCMDz@SyzygyInc.com      ←Authorized to send mail from this LPAR
CC: JOBLOG@SyzygyInc.com         ←send copy to JOBLOGs
*
```

INCDSN

data.set.name | data.set.name(member)

The INCDSN parameter allows the site to specify a default dataset (or dataset and member, which is like performing INCDSN and INCMEM together) which will contain the command scripts that will be included via the ++INCLUDE command. If not specified, the CMDDSN dataset (or //IEFRDER DD if provided) will be used.

NOTE: Tape data sets and migrated data sets are not supported.

Example:

```
........
* This is an example of SYZCMD00 Startup Parameter member
CMDDSN=SYS1.COMMANDS
************************************************
INCDSN=SYS1.commands
************************************************
SYSEXEC=USER.CLIST
STARTUP=QUIET
*CMDMEM=NOTHING
*INCMEM=NOTHING
RACF=YES
SYSOUT=F
SYZMAIL=Yes                   ←Allow email Nicknames to be generated
MAXMSG=100                    ←Default is 50
TZ=-0700                      ←Time zone is Pacific time (-700)
To: AllMail@SyzygyInc.com     ←Default to use if script forgets or is invalid
From: SyzCMDz@SyzygyInc.com   ←Authorized to send mail from this LPAR
CC: JOBLOG@SyzygyInc.com      ←send copy to JOBLOGs
*
```

INCMEM

membername

The INCMEM parameter allows the site to specify a default member (of the provided INCDSN dataset), of one (1) to eight (8) characters which will contain the command script to be used for the ++INCLUDE directive to be executed. This parameter can be specified in the EXECution parm of the SyzCMD/z task instead.

NOTE: Only one member is allowed in the parmlib defaults dataset, but any number can be used in the execution parms separated by “;”s.

Example:

```
........
* This is an example of SYZCMD00 Startup Parameter member
CMDDSN=SYS1.COMMANDS
INCDSN=SYS1.commands
SYSEXEC=USER.CLIST
STARTUP=QUIET
CMDMEM=DEFAULT
************************************************
INCMEM=INCLUDE
************************************************
RACF=YES
SYSOUT=F
SYZMAIL=Yes                  ←Allow email Nicknames to be generated
MAXMSG=100                   ←Default is 50
TZ=-0700                     ←Time zone is Pacific time (-700)
To: AllMail@SyzygyInc.com    ←Default to use if script forgets or is invalid
From: SyzCMDz@SyzygyInc.com  ←Authorized to send mail from this LPAR
CC: JOBLOG@SyzygyInc.com     ←send copy to JOBLOGs
*
```

INTERVAL

nnnn[S|M|H]

Description:

SyzCMD/z has many sub-commands which will wait for something to happen. “WAIT until” and also the Started and stopped commands are good examples of this. The default “interval” time of these waits is 15 seconds. Many times, it is a waste of time to wait for 15 seconds for something that may occur in a second or two and it was discovered that the system resources involved in reducing the wait interval to even 1 second was trivial. You can, with this command set that interval to any value from 1 second up to 9999

Parameters:

nnn [S|M|H]

nnnn - required - Up to a 4 digit number of seconds (the default), minutes or hours to wait to allow syzCMD/z to re-check the environment. The maximum allowed value of digits is 9999. “S” is the default and sets the preceding number to be that many seconds. You can also specify “M” for minutes or H for hours. If not specified, the default is 15 seconds (the SyzCMD/z default)

Example:

```
........
* This is an example of SYZCMD00 Startup Parameter member
CMDDSN=SYS1.COMMANDS
INCDSN=SYS1.commands
SYSEXEC=USER.CLIST
STARTUP=QUIET
CMDMEM=DEFAULT
RACF=YES
INTERVAL=5                    ← Make sure we scan every two seconds
SYSOUT=F
SYZMAIL=Yes                   ←Allow email Nicknames to be generated
MAXMSG=100                    ←Default is 50
TZ=-0700                      ←Time zone is Pacific time (-700)
```

```
To: AllMail@SyzygyInc.com           ←Default to use if script forgets or is invalid
From: SyzCMDz@SyzygyInc.com         ←Authorized to send mail from this LPAR
CC: JOBLOG@SyzygyInc.com            ←send copy to JOBLOGs
*
```

LOGDSN

data.set.name | data.set.name(member)

The LOGDSN startup parameter is specified in the SYS1.PARMLIB startup dataset for SyzCMD/z and allows the site to specify a default dataset (or dataset and member which will be used by the LOG= command (described later in this manual). If the running task specifies a //LOGDSN DD, it will override this specification. However, using this parameter will allow you to eliminate the //LOGDSN DD completely from your task JCL. If specified in PARMLIB, the dataset must already exist.

NOTE: Tape data sets and migrated data sets are not supported.

Example:

```
........
* This is an example of SYZCMD00 Startup Parameter member
************************************************
CMDDSN=SYS1.COMMANDS
************************************************
INCDSN=SYS1.commands
SYSEXEC=USER.CLIST
STARTUP=QUIET
*CMDMEM=NOTHING
*INCMEM=NOTHING
LOGDSN=SYS1.JOBLOG
RACF=YES
SYSOUT=F
SYZMAIL=Yes                      ←Allow email Nicknames to be generated
MAXMSG=100                       ←Default is 50
TZ=-0700                         ←Time zone is Pacific time (-700)
To: AllMail@SyzygyInc.com        ←Default to use if script forgets or is invalid
From: SyzCMDz@SyzygyInc.com      ←Authorized to send mail from this LPAR
CC: JOBLOG@SyzygyInc.com         ←send copy to JOBLOGs
*
```

MAXWAIT

9999 H (hrs), M(minutes), S(seconds) | FOREVER (forever is the default)

The "MAXWAIT" GLOBAL setting parameter is used to tell SyzCMD/z the absolute maximum amount of time to "WAIT" for something to happen. For instance, the site may have coded a STARTED (or WAIT UNTIL STARTED) command to wait for some task to start, but in some cases if the task has not started by a certain amount of time (say 5 minutes), the site may wish to take some other action. Otherwise SyzCMD/z would merely issue the normal message to the console operator that lets them know we are waiting, and then proceeded to wait forever until it happens. This GLOBAL command can also be used as a script command so that you can set different wait times based on different events.

When used as a script command it has effect on certain IF related commands (i.e. IF WTOR) where it can set a maximum amount of time to wait for the console operator to respond to the request or IF STARTED or IF STOPPED where it can end the wait after a certain amount of time to be retried or just continued.

The default is "FOREVER".
Parameters:

9999 H|M|S	(H= hours, M=Minutes, S=Seconds
FOREVER	(default, wait forever)

Example:

.....SYS1.PARMLIB(SYZCMD00) member contents...

```
CMDDSN=SYS1.COMMANDS
INCDSN=SYS1.commands
SYSEXEC=USER.CLIST
STARTUP=QUIET
*CMDMEM=NOTHING
*INCMEM=NOTHING
RACF=YES
MAXWAIT=FOREVER
SYSOUT=F
DEBUG=NO
STATISTICS=Default
SYZMAIL=Yes                     Allow email Nicknames to be generated
MAXMSG=100                      Default is 50
TZ=-0700                        Time zone is Pacific time (-700)
To: AllMail@SyzygyInc.com       Default to use if script forgets or is invalid
From: SyzCMDz@SyzygyInc.com     Authorized to send mail from this LPAR
CC: JOBLOG@SyzygyInc.com        send copy to JOBLOGs
.....
```

RACF

Yes|No

The RACF parameter allows the site to specify whether or not SyzCMD/z should ask RACF to verify (via the FACILITY class resource of SYZCMDZ) to authorize the use of SyzCMD/z for command execution. The default is YES.

Example:

```
........
* This is an example of SYZCMD00 Startup Parameter member
CMDDSN=SYS1.COMMANDS
INCDSN=SYS1.commands
SYSEXEC=USER.CLIST
STARTUP=QUIET
*CMDMEM=NOTHING
*INCMEM=NOTHING
RACF=YES
SYSOUT=F
SYZMAIL=Yes                  ←Allow email Nicknames to be generated
MAXMSG=100                   ←Default is 50
TZ=-0700                     ←Time zone is Pacific time (-700)
To: AllMail@SyzygyInc.com    ←Default to use if script forgets or is invalid
From: SyzCMDz@SyzygyInc.com  ←Authorized to send mail from this LPAR
CC: JOBLOG@SyzygyInc.com     ←send copy to JOBLOGs
*
```

SIMULATE

Yes|No (or anything but Yes will result in No)

The SIMULATE parmlib control command is used to cause SyzCMD/z to set the default mode of operation of SyzCMD/z to “simulate” (or not) the issuing of commands and replies to the system. The site can use this command to turn the Simulation Mode of SyzCMD/z on via “SIMUALTE=Yes” or off via “SIMULATE=No”.

Parameters:
Yes | No

Example:

```
........
* This is an example of SYZCMD00 Startup Parameter member
CMDDSN=SYS1.COMMANDS
INCDSN=SYS1.commands
SYSEXEC=USER.CLIST
STARTUP=QUIET
*CMDMEM=NOTHING
*INCMEM=NOTHING
SIMULATE=No
RACF=YES
SYSOUT=F
SYZMAIL=Yes                      ←Allow email Nicknames to be generated
MAXMSG=100                       ←Default is 50
TZ=-0700                         ←Time zone is Pacific time (-700)
To: AllMail@SyzygyInc.com        ←Default to use if script forgets or is invalid
From: SyzCMDz@SyzygyInc.com      ←Authorized to send mail from this LPAR
CC: JOBLOG@SyzygyInc.com         ←send copy to JOBLOGs
*
........
```

STARTUP

Quiet

The STARTUP parameter allows the site to specify whether or not SyzCMD/z should display the startup messages which normally state the product name, license information and the number of days left under the license to the system console.

NOTE: If QUIET is specified, and the site gets within 30 days of expiration of their license, the expiration soon messages will still be issued

Example:

```
........
* This is an example of SYZCMD00 Startup Parameter member
CMDDSN=SYS1.COMMANDS
INCDSN=SYS1.commands
SYSEXEC=USER.CLIST
STARTUP=QUIET
*CMDMEM=NOTHING
*INCMEM=NOTHING
RACF=YES
SYSOUT=F
SYZMAIL=Yes                     ←Allow email Nicknames to be generated
MAXMSG=100                      ←Default is 50
TZ=-0700                        ←Time zone is Pacific time (-700)
To: AllMail@SyzygyInc.com       ←Default to use if script forgets or is invalid
From: SyzCMDz@SyzygyInc.com     ←Authorized to send mail from this LPAR
CC: JOBLOG@SyzygyInc.com        ←send copy to JOBLOGs
*
```

STEPLIB

No|Yes (or anything but No)

The STEPLIB parameter allows the site to specify that SyzCMD/z should not be allowed to execute if the program is loaded from a //STEPLIB or that a //STEPLIB exists in the step which executes the SyzCMD/z program. The default is “Yes” which will result in SyzCMD/z being allowed to execute with a STEPLIB, (as opposed to “NO” which would allow execution via LINKLIST only).

Example:

```
........
* This is an example of SYZCMD00 Startup Parameter member
CMDDSN=SYS1.COMMANDS
INCDSN=SYS1.commands
SYSEXEC=USER.CLIST
STARTUP=QUIET
*CMDMEM=NOTHING
*INCMEM=NOTHING
RACF=YES
SYSOUT=F
STEPLIB=OK
SYZMAIL=Yes                     ←Allow email Nicknames to be generated
MAXMSG=100                      ←Default is 50
TZ=-0700                        ←Time zone is Pacific time (-700)
To: AllMail@SyzygyInc.com       ←Default to use if script forgets or is invalid
From: SyzCMDz@SyzygyInc.com     ←Authorized to send mail from this LPAR
CC: JOBLOG@SyzygyInc.com        ←send copy to JOBLOGs
*
```

STOPCHECK | STOPWAIT

No|Yes|9999 H|M|S

The STOPCHECK parameter tells SyzCMD/z whether or not to recheck that a task is still stopped at the end of the original checking. Since mainframes can be fairly fast and have lots of tasks starting and stopping, it was found that it was possible that a task TASKA could be running (or even not active at all), but between the time that SyzCMD/z checks and sees that TASKA is inactive, and the time that the next script command is processed, TASKA could be running again. A good example of this is a case where there is a JOB TASKA in an initiator, and another job also TASKA, is waiting to start. TASKA could finish, SyzCMD/z sees that it is gone and moves on to the next script line, but the next TASKA is now again running. By default, SyzCMD/z waits 1 second, and then rechecks to see if the task is still inactive, and if not, it continues the wait. Instead of 1 second, the site can set any amount of time, up to 9999Hours. It is rarely necessary to use any time at all, since 1 second is normally enough to keep job stacking from being an issue. This same process is used when the
WAIT UNTIL STOPPED 9999h|m|s
command is use, (i.e. there is a delay of 1s by default and then it rechecks), and this parameter will affect that logic as well.

Example:

```
STOPCHECK = 4Seconds    (pause for 4 sec and recheck task)
........
* This is an example of SYZCMD00 Startup Parameter member
CMDDSN=SYS1.COMMANDS
INCDSN=SYS1.commands
SYSEXEC=USER.CLIST
STARTUP=QUIET
*CMDMEM=NOTHING
*INCMEM=NOTHING
RACF=YES
SYSOUT=F
STEPLIB=OK
STOPCHECK=No                  1second is the default (turn it off)
SYZMAIL=Yes                   Allow email Nicknames to be generated
MAXMSG=100                    Default is 50
TZ=-0700                      Time zone is Pacific time (-700)
To: AllMail@SyzygyInc.com     Default to use if script forgets or is invalid
From: SyzCMDz@SyzygyInc.com   Authorized to send mail from this LPAR
CC: JOBLOG@SyzygyInc.com      send copy to JOBLOGs
*
```

SYSEXEC

data.set.name | Data.set.name(member)

The SYSEXEC parameter allows the site to specify a CLIST or REXX exec default dataset (or dataset and member, which is like performing SYSEXEC and prefixing the REXX exec name together) which will contain the REXX execs that can be used by SyzCMD/z. If the running task specifies a //SYSEXEC DD, it will override this specification. However, using this parameter will allow you to eliminate the //SYSEXEC DD completely from your task JCL.

NOTE: Tape data sets and migrated data sets are not supported.

Example:

```
........
* This is an example of SYZCMD00 Startup Parameter member
CMDDSN=SYS1.COMMANDS
INCDSN=SYS1.commands
**************************************************
SYSEXEC=USER.CLIST
**************************************************
STARTUP=QUIET
*CMDMEM=NOTHING
*INCMEM=NOTHING
RACF=YES
SYSOUT=F
SYZMAIL=Yes                     ←Allow email Nicknames to be generated
MAXMSG=100                      ←Default is 50
TZ=-0700                        ←Time zone is Pacific time (-700)
To: AllMail@SyzygyInc.com       ←Default to use if script forgets or is invalid
From: SyzCMDz@SyzygyInc.com     ←Authorized to send mail from this LPAR
CC: JOBLOG@SyzygyInc.com        ←send copy to JOBLOGs
*
```

SYSOUT

Class

The SYSOUT parameter allows the site to specify the class that SyzCMD/z should use to allocate any SYSOUT datasets (for REXX exec uses or normal operation). The default is Class “A”.

Example:

```
........
* This is an example of SYZCMD00 Startup Parameter member
CMDDSN=SYS1.COMMANDS
INCDSN=SYS1.commands
SYSEXEC=USER.CLIST
STARTUP=QUIET
*CMDMEM=NOTHING
*INCMEM=NOTHING
RACF=YES
SYSOUT=F
SYZMAIL=Yes                     ←Allow email Nicknames to be generated
MAXMSG=100                      ←Default is 50
TZ=-0700                        ←Time zone is Pacific time (-700)
To: AllMail@SyzygyInc.com       ←Default to use if script forgets or is invalid
From: SyzCMDz@SyzygyInc.com     ←Authorized to send mail from this LPAR
CC: JOBLOG@SyzygyInc.com        ←send copy to JOBLOGs
*
```

To:

Email address or Nickname or &variable

The "To:" GLOBAL setting parameter is used to tell SyzCMD/z 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: &notify", 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 SyzCMD/z script. **This GLOBAL command can also be used as a script command so that you can set different values based on different events.**

(i.e. TO: SCRIPT_Errors@thissite.com .

Parameters:

Email Address or Nickname or &Variable

Example:

.....SYS1.PARMLIB(SYZCMD00) member contents...

```
CMDDSN=SYS1.COMMANDS
INCDSN=SYS1.commands
SYSEXEC=USER.CLIST
STARTUP=QUIET
*CMDMEM=NOTHING
*INCMEM=NOTHING
RACF=YES
SYSOUT=F
SYZMAIL=Yes                    ←Allow email Nicknames to be generated
MAXMSG=100                     ←Default is 50
TZ=-0700                       ←Time zone is Pacific time (-700)
To: AllMail@SyzygyInc.com      ←Default to use if script forgets or is invalid
From: SyzCMDz@SyzygyInc.com    ←Authorized to send mail from this LPAR
CC: JOBLOG@SyzygyInc.com       ←send copy to JOBLOGs
.....
```

From:

Email address or Nickname or &variable

The "From:" GLOBAL setting parameter is used to tell SyzCMD/z 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: &notify", 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 SyzCMD/z script. **This GLOBAL command can also be used as a script command so that you can set different values based on different events.**

(i.e. From: SyzCMDz@thissite.com .

Parameters:

Email Address or Nickname or &Variable

Example:

.....SYS1.PARMLIB(SYZCMD00) member contents...

```
CMDDSN=SYS1.COMMANDS
INCDSN=SYS1.commands
SYSEXEC=USER.CLIST
STARTUP=QUIET
*CMDMEM=NOTHING
*INCMEM=NOTHING
RACF=YES
SYSOUT=F
SYZMAIL=Yes                  ←Allow email Nicknames to be generated
MAXMSG=100                   ←Default is 50
TZ=-0700                     ←Time zone is Pacific time (-700)
To: AllMail@SyzygyInc.com    ←Default to use if script forgets or is invalid
From: SyzCMDz@SyzygyInc.com  ←Authorized to send mail from this LPAR
```

CC: JOBLOG@SyzygyInc.com ←send copy to JOBLOGs
.....

ReplyTo:

Email address or Nickname or &variable

The "ReplyTo:" GLOBAL setting parameter is used to tell SyzCMD/z what default email address or nickname (nicknames are described in the SyzMAIL/z manual) to use for all outgoing eMail messages. The ReplyTo parameter will override the "FROM" parameter when the recipient replies to the email. Instead of the reply being sent to the "From" ID, it will go to the ReplyTo ID. This setting is not required. This can be set to a variable such as "ReplyTo: &notify", which will send the email or SMS text message with a ReplyTo of 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 SyzCMD/z script. **This GLOBAL command can also be used as a script command so that you can set different values based on different events.**

(i.e. ReplyTo: SyzCMDz@thissite.com .

Parameters:

Email Address or Nickname or &Variable

Example:

.....SYS1.PARMLIB(SYZCMD00) member contents...

```
CMDDSN=SYS1.COMMANDS
INCDSN=SYS1.commands
SYSEXEC=USER.CLIST
STARTUP=QUIET
*CMDMEM=NOTHING
*INCMEM=NOTHING
RACF=YES
SYSOUT=F
SYZMAIL=Yes                   ←Allow email Nicknames to be generated
MAXMSG=100                    ←Default is 50
TZ=-0700                      ←Time zone is Pacific time (-700)
To: AllMail@SyzygyInc.com     ←Default to use if script forgets or is invalid
ReplyTo: SyzCMDz@SyzygyInc.com←Authorized to receive back from recipient
CC: JOBLOG@SyzygyInc.com      ←send copy to JOBLOGs
.....
```

Cc:

Email address or Nickname or &variable

The “Cc:” GLOBAL setting parameter is used to tell SyzCMD/z 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 code a default carbon copy address to receive a CC of the email. This can be set to a variable such as “Cc: &notify”, 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 can be handy in cases where you want copies of all outgoing mail to be sent to a “log” address. **This GLOBAL command can also be used as a script command so that you can set different values based on different events.**

(i.e. Cc: JOBLOG@thissite.com .

Parameters:

Email Address or Nickname or &Variable

Example:

.....SYS1.PARMLIB(SYZCMD00) member contents...

```
CMDDSN=SYS1.COMMANDS
INCDSN=SYS1.commands
SYSEXEC=USER.CLIST
STARTUP=QUIET
*CMDMEM=NOTHING
*INCMEM=NOTHING
RACF=YES
SYSOUT=F
SYZMAIL=Yes                    ←Allow email Nicknames to be generated
MAXMSG=100                     ←Default is 50
TZ=-0700                       ←Time zone is Pacific time (-700)
To: AllMail@SyzygyInc.com      ←Default to use if script forgets or is invalid
From: SyzCMDz@SyzygyInc.com    ←Authorized to send mail from this LPAR
CC: JOBLOG@SyzygyInc.com       ←send copy to JOBLOGs
.....
```

TimeZone or TZ

Time zone offset to use (default = PST)

The “TimeZone or TZ” GLOBAL setting parameter is used to tell SyzCMD/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(SYZCMD00) member contents...

```
CMDDSN=SYS1.COMMANDS
INCDSN=SYS1.commands
SYSEXEC=USER.CLIST
STARTUP=QUIET
*CMDMEM=NOTHING
*INCMEM=NOTHING
RACF=YES
SYSOUT=F
SYZMAIL=Yes                       ←Allow email Nicknames to be generated
MAXMSG=100                        ←Default is 50
TZ=-0700                          ←Time zone is Pacific time (-700)
To: AllMail@SyzygyInc.com         ←Default to use if script forgets or is invalid
From: SyzCMDz@SyzygyInc.com       ←Authorized to send mail from this LPAR
CC: JOBLOG@SyzygyInc.com          ←send copy to JOBLOGs
.....
```

STATistics

All | Verbose or NONE or DEFAULT

The “STATistics” GLOBAL setting parameter is used to tell SyzCMD/z what level of JOB end statistics to show. This is VERY similar to the old EOSM command (which is still supported but seldome used). Setting this parameter will control whetheror not SyzCMD/z execution statistics are displayed, and to what level of detail they are shown. ALL or Verbose is the maximum level of detail.

Parameters:

All | Verbose	:	**Show All statistics**
NONE	:	**Show NO statistics**
DEFAULT	:	**Show “normal” statistics**

Example:

.....SYS1.PARMLIB(SYZCMD00) member contents...

```
CMDDSN=SYS1.COMMANDS
INCDSN=SYS1.commands
SYSEXEC=USER.CLIST
STARTUP=QUIET
*CMDMEM=NOTHING
*INCMEM=NOTHING
RACF=YES
SYSOUT=F
STATISTICS=Default
SYZMAIL=Yes                   ←Allow email Nicknames to be generated
MAXMSG=100                    ←Default is 50
TZ=-0700                      ←Time zone is Pacific time (-700)
To: AllMail@SyzygyInc.com     ←Default to use if script forgets or is invalid
From: SyzCMDz@SyzygyInc.com   ←Authorized to send mail from this LPAR
CC: JOBLOG@SyzygyInc.com      ←send copy to JOBLOGs
.....
```

SUBJect: or Subj:

Any text (including &variables)

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: &LPAR from &TASKNAME (&TASKID)” (results in a subject in the email of: PRODA 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:

.....SYS1.PARMLIB(SYZCMD00) member contents...

```
CMDDSN=SYS1.COMMANDS
INCDSN=SYS1.commands
SYSEXEC=USER.CLIST
STARTUP=QUIET
*CMDMEM=NOTHING
*INCMEM=NOTHING
RACF=YES
SYSOUT=F
SYZMAIL=Yes                      ←Allow email Nicknames to be generated
MAXMSG=100                       ←Default is 50
TZ=-0700                         ←Time zone is Pacific time (-700)
To: AllMail@SyzygyInc.com        ←Default to use if script forgets or is invalid
From: SyzCMDz@SyzygyInc.com      ←Authorized to send mail from this LPAR
CC: JOBLOG@SyzygyInc.com         ←send copy to JOBLOGs
Subj: Message from &TASKNAME     ←Default Subject to use if none provided
.....
```

SYSMAIL

Yes | No

The “SYZMAIL” GLOBAL setting parameter is used to tell SyzCMD/z whether or not the SYZMAIL/z product is installed at this site and should be used to send email or if SyzCMD/z should generate it’s own email messages directly to SMTP. There are several advantages to using SyzMAIL/z, including the ability to use nicknames that will be resolved dynamically by SyzMAIL/z. Otherwise (for nicknames) the script must contain all of the detail information used to generate the email.

Parameters:

Yes | No (NO is the default)

Example:

.....SYS1.PARMLIB(SYZCMD00) member contents...

```
CMDDSN=SYS1.COMMANDS
INCDSN=SYS1.commands
SYSEXEC=USER.CLIST
STARTUP=QUIET
*CMDMEM=NOTHING
*INCMEM=NOTHING
RACF=YES
SYSOUT=F
STATISTICS=Default
SYZMAIL=Yes                     ←Allow email Nicknames to be generated
MAXMSG=100                      ←Default is 50
TZ=-0700                        ←Time zone is Pacific time (-700)
To: AllMail@SyzygyInc.com       ←Default to use if script forgets or is invalid
From: SyzCMDz@SyzygyInc.com     ←Authorized to send mail from this LPAR
CC: JOBLOG@SyzygyInc.com        ←send copy to JOBLOGs
.....
```

MAXMSG

Nnnn (default is 50 lines)

The “MAXMSG” GLOBAL setting parameter is used to tell SyzCMD/z how many lines the maximum email message will contain. If this number is exceeded, the recipient will receive a message that the email “may” have been truncated. The Default is 50 lines. The maximum number of lines is 9999. **This GLOBAL command can also be used as a script command so that you can set different values based on different events.**

Parameters:

nnnn : Max of 9999 lines (default is 50 lines)

Example:

.....SYS1.PARMLIB(SYZCMD00) member contents...

```
CMDDSN=SYS1.COMMANDS
INCDSN=SYS1.commands
SYSEXEC=USER.CLIST
STARTUP=QUIET
*CMDMEM=NOTHING
*INCMEM=NOTHING
RACF=YES
SYSOUT=F
STATISTICS=Default
SYZMAIL=Yes                      ←Allow email Nicknames to be generated
MAXMSG=100                       ←Default is 50
TZ=-0700                         ←Time zone is Pacific time (-700)
To: AllMail@SyzygyInc.com        ←Default to use if script forgets or is invalid
From: SyzCMDz@SyzygyInc.com      ←Authorized to send mail from this LPAR
CC: JOBLOG@SyzygyInc.com         ←send copy to JOBLOGs
.....
```

DEBUG

Yes | No (no is the default)

The "DEBUG" GLOBAL setting parameter is used to tell SyzCMD/z whether or not to display DEBUGing information on the console and in the LOG. This is normally only to be used at the direction of Syzygy personnel, but the site may use it if they are testing out features and wish to know detailed information on what is being performed for them within the script. The default is "NO". **This GLOBAL command can also be used as a script command so that you can set different values based on different events.**

Parameters:

Yes | No (no is the default)

Example:

.....SYS1.PARMLIB(SYZCMD00) member contents...

```
CMDDSN=SYS1.COMMANDS
INCDSN=SYS1.commands
SYSEXEC=USER.CLIST
STARTUP=QUIET
*CMDMEM=NOTHING
*INCMEM=NOTHING
RACF=YES
SYSOUT=F
DEBUG=NO
STATISTICS=Default
SYZMAIL=Yes                      ←Allow email Nicknames to be generated
MAXMSG=100                       ←Default is 50
TZ=-0700                         ←Time zone is Pacific time (-700)
To: AllMail@SyzygyInc.com        ←Default to use if script forgets or is invalid
From: SyzCMDz@SyzygyInc.com      ←Authorized to send mail from this LPAR
CC: JOBLOG@SyzygyInc.com         ←send copy to JOBLOGs
.....
```

VERBOSE

Yes | No (no is the default)

The “VERBOSE” GLOBAL setting parameter is used to tell SyzCMD/z whether or not to display VERBOSE (longer) messages for most processes including WAIT messages. The VERBOSE messages normally show additional options that may be available to the console operator during a WAIT for some resource. The default is “NO”. **This GLOBAL command can also be used as a script command so that you can set different values based on different events.**

Parameters:

Yes | No (no is the default)

Example:

.....SYS1.PARMLIB(SYZCMD00) member contents...

```
CMDDSN=SYS1.COMMANDS
INCDSN=SYS1.commands
SYSEXEC=USER.CLIST
STARTUP=QUIET
*CMDMEM=NOTHING
*INCMEM=NOTHING
RACF=YES
SYSOUT=F
VERBOSE=NO
STATISTICS=Default
SYZMAIL=Yes                  ←Allow email Nicknames to be generated
MAXMSG=100                   ←Default is 50
TZ=-0700                     ←Time zone is Pacific time (-700)
To: AllMail@SyzygyInc.com    ←Default to use if script forgets or is invalid
From: SyzCMDz@SyzygyInc.com  ←Authorized to send mail from this LPAR
CC: JOBLOG@SyzygyInc.com     ←send copy to JOBLOGs
.....
```

WTOPREFix

Yes | No (yes is the default)

The “WTOPREFix” GLOBAL setting parameter is used to tell SyzCMD/z whether or not to display the “* taskname *” in front of WTO command text and “*!taskname!*” in front of WTOR command textThe default is “YES”. This GLOBAL command can also be used as a script command so that you can set different values based on different events.

Parameters:

Yes | No (no is the default)

Example:

```
.....SYS1.PARMLIB(SYZCMD00) member contents...

CMDDSN=SYS1.COMMANDS
INCDSN=SYS1.commands
SYSEXEC=USER.CLIST
STARTUP=QUIET
*CMDMEM=NOTHING
*INCMEM=NOTHING
RACF=YES
SYSOUT=F
VERBOSE=NO
WTOPREF=YES                    ← Add taskname prefix to all WTO and WTOH
STATISTICS=Default
SYZMAIL=Yes                    □Allow email Nicknames to be generated
MAXMSG=100                     □Default is 50
TZ=-0700                       □Time zone is Pacific time (-700)
To: AllMail@SyzygyInc.com      □Default to use if script forgets or is invalid
From: SyzCMDz@SyzygyInc.com    □Authorized to send mail from this LPAR
CC: JOBLOG@SyzygyInc.com       □send copy to JOBLOGs
.....
```

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                                              *
**************************************************
.....
```

::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 SyzCMD/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=4                                                    |
ELSE                                                      |
  Do some other commands                                  |
ENDIF                                                     |
EXIT                                                      |
::CICS2                <<<------labelname = CICS2----------
IF STARTED CICS002
EXIT=4
ELSE
  Do some other commands
ENDIF
EXIT
.....
```

++INCLUDE

SEQUENTIAL.DATA.SET
PDS.DATA.SET(member)
NAME (Command script name from currently allocated //IEFRDER)

The ++INCLUDE control command is used to include a script member from the currently allocated INCDSN, or from CMDDSN (if INCDSN is not defined) or from the //INTRDER DD (if neither INCDSN or CMDDSN is defined and //IEFRDER is allocated as a PDS). If one of those requirements is not true, then the command is failed.   You may define the INCDSN on the ++INCLUDE script line. The command script is loaded and executed, once it is finished, command execution restarts where it left off in the original command script.

Parameters:
A Sequential or PDS(member) dataset or a 1 to 8 character command name to include is required or may be defaulted to from a previous setting

Example:

```
........
*  See which CICS is up
++INCLUDE=WHATCICS
...   the WHATCICS command script is executed then returns to the next line
++INCLUDE TEST.PDS(TESTME)
... the TEST.PDS dataset is located and the TESTME member is executed
IF REXXRC > 4                 **  Was return code 5 or more?
  $P PRT15
  $PI
  PAUSE 5s
  $TI1-5,ABCD
  $SI1-5
ELSE
* Less than 5 (4 or less)
   $PI
   PAUSE 5s
   $TI1-5,CDEF
   $SI1-5
ENDIF
.....
```

5. Variables (both #Dynamic and &Static)

There are two types of Variables which exist under SyzCMD/z, Static Variables, which begin with a "&" (Ampersand) and are things that are set to static items, time related; hours, minutes seconds, date related; Month, day year, day of week, etc. task related; taskname, stepname, jobid, RACF job owner, notify on the jobcard, current and maximum condition codes, etc. Then there are Dynamic variables, which begin with a "#" USA pound sign, and can be literally anything you want or need them to be, they can be called anything up to 10 characters in length that begins with a "#" like "#A" or #whatisthis. Whatever name you call them, will be translated to upper case, so #var001 and #VAR001 are exactly the same dynamic variable. The Dynamic variable can be up to 50 characters long and contain embedded blanks and can be made up of static variables. For instance, if you set a dynamic variable called "#today" to "Now is &LDOW, &MN/&DD/20&YY at &HH:&MM:&SS" and it just happens to be Monday, November 23, 2022 and 06:15:00 am, then the dynamic variable #today will be "Now is Monday 11/23/2022 at 06:15:00" (minus the quotes).

These two types of variables are quite useful and each have their proper use. Static variables are great for comparing against and using in messages because they are quantifiable, but Dynamic variables excel at being used for more esoteric things, like building sentences, counting, keeping track of items and a myriad of other things. The big difference is that Dynamic variables can contain any character and can be quite long, whereas static variables tend to be relatively short and NEVER contain any spaces. The important thing to remember about spaces is that if you are going to use them in a Dynamic Variable, you have to surround the data with some means of telling that the blanks are supposed to "be there". Quotes (')work quite well for this, and several other special characters are also allowed, just remember that if you start the string with a single quote ('), you should also end with one, don't mix the single (')and double quotes (") up because one of them (usually the second one) will either be ignored, and the variable will simply be whatever the string was up to the first space, or might be the whole string with the double quotes on the end. You also can't mix their use, so if you have a string the you want to contain a single quote like "it's going to be a nice day", then surround the string with double quotes '"', which is not the same as just doubling up your quotes (' ').

Use of variables is quite powerful, but you have to be consistent or you will end up with meaningless strings of information.

At 11:23 am, &HH:MM will end up as "11:MM" because the static variable for minutes is &MM not just MM.

&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 - 2 digits, Hour of the current Time of Day (24 Hour Clock)
&MM - 2 digits, Minutes of current Time of Day
&SS - 2 digits, Seconds of the current Time of Day
&TT - 2 digits, Tenths and hundredths of a second
&HHMM - 4 digits, Current Hours and Minutes 0123
&HH:MM - 6 characters, Current Hours and Minutes 01:23
&HHMMSS - 6 digits, Current Hours, Minutes and Seconds 102345
&HH:MM:SS - 8 characters, Current Hours, Minutes and Seconds 01:23:45
&TIME - 8 characters, Current Hours, Minutes and Seconds 01:23:45

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 - 2 digits, (not MM) Current Month
&DD - 2 digits, Current Day
&YY - 2 digits, Current Year
&YYYY - 4 digits, current year
&DDD- 3 digits, Julian DAY
&YYDDD - 5 digits, current Julian date
&YYYYDDD - 7 digits, current Julian date
&MNDD - 4 digits. Current Month and Day digits
&MMDD - 4 digits. Current Month and Day digits
&SDATE - 4 digits. Current Month and Day digits
&MMDDYY -6 digits, Current Month, Day and Year digits
&MNDDYY - 6 digits, Current Month, Day and Year digits
&MM/DD - 5 digits, Current Month and Day (with '/' separator)
&MN/DD - 5 digits, Current Month and Day (with '/' separator)
&SDATE - 5 digits, Current Month and Day (with '/' separator)
&DATE - 8 characters, Current Month and Day (with '/' separator)
&MM/DD/YY - 8 characters, Current Month, day and year (with '/' separator)
&MN/DD/YY - 8 characters, Current Month, day and year (with '/' separator)

Example:

....
WTO The current Date is &MN/DD/YY ←-note Month is &MN when used singly, not &MM (because &MM = minutes :)
**** Results in the following WTO:
"The current date is 11/DD/YY" ← because we forgot to use &DD and &YY

.....

&DAY - Day of Week STATIC variable:

&DAY- (up to) Nine Character Day of Week

(MONDAY-TUESDAY-WEDNESDAY-THURSDAY-FRIDAY-SATURDAY-SUNDAY)

Example:

```
........
WTO The current Day is &DAY

**** Results in the following WTO:

"The current Day is MONDAY"  (sans quotes)

.....
```

&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”  (sans quotes)

.....
```

&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”

.....
```

&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" (sans quotes)

.....

&MONTH - Full Month STATIC variable:

&MONTH - (up to) Nine Character MONTH
(JANUARY-DECEMBER)

Example:

....
WTO The current Month is &MONTH

**** Results in the following WTO:

"The current Month is NOVEMBER" (sans quotes)

.....

&ASID - Current Addresspace ID STATIC variable:

&ASID - 4 character Addresspace ID

Example:

```
........
WTO The current ASID is &ASID
```

**** Results in the following WTO:

“The current ASID is 0023”

.....

&ASIDX - Current Hexidecimal Addresspace ID STATIC variable:

&ASID - 4 character Hexidecimal Addresspace ID

Example:

```
........
WTO The current Hex ASID is &ASIDX (decimal &ASID)

**** Results in the following WTO:

"The current ASID is 0017 (decimal 23)"    (Sans quotes)

.....
```

&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"

…..

&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"

.....

&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**"

.....

&MAXRESult- Maximum task result:

&maxresult - will be filled with the highest result (so far) and will be one of the following:
Ended
Flushed
Abended
Active (still running)

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**"

.....

&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- MSGID of the Message that started this task:

&msgID - 1-8 character MSGID from the SyzCMD/z MPF entry
That started this task.

****Note SyzMPF/z is a much better facility to process console messages, but SyzCMD/z does have limited capability in starting scripts based on MPF entries.

Example:
.... task started by MPF entry for msgid IEF0997I
WTO The current MSGID is &MSGID
**** Results in the following WTO:
"The current MSGID is **IEF0997I**"

.....

&MSGNAME- Name of the task issuing (THIS) Message that started this task:

&msgname - 1-8 character taskname from the SyzCMD/z MPF entry
That started this task.

****Note SyzMPF/z is a much better facility to process console messages, but SyzCMD/z does have limited capability in starting scripts based on MPF entries.

Example:
.... task started by MPF entry for msgid IEF0997I
WTO The current message is &MSGID from &MSGNAME
**** Results in the following WTO:
"The current message is IEF0997I from **MYTASK1**"

.....

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

&NOTIFY - 1 – 8 character NOTIFY= parm name

Example:

……..
WTO The current tasks NOTIFY=&notify
**** Results in the following WTO:
"The current tasks NOTIFY=**BRIANW**"

…..

&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**"

.....

&RACFGRP- RACF group of this task:

&racfgrp	-	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**”

…..

&RESPONSE- The most recent RESPONSE text received:

&response - 1 –23 character (most current) Response text received

(this field is set by either the ASKOPER or IFWTOR control commands. It is the response received to the MOST RECENT of those two control commands)

Example:

```
........
WTO The last Response received was=&REPONSE
**** Results in the following WTO:
“The last Response received was=YES”

.....
```

&REXXRC- The most recent REXX return code received:

&rexxrc - 1 – 4 character (most current) REXX return code received

Example:

....
WTO The last REXX command received a CC=&REXXRC return code
**** Results in the following WTO:
"The last REXX command received a CC=12 return code"

.....

&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**”

.....

&STCID - The current Started TASK ID name

&stcid - up to 8 character taskid

Example:

```
.....
EOSM=YES
ECHO=ALL
* it’s a shutdown for IPL so make sure we were started as a sub=mstr space
*  they may have just said “S SHUTDOWN” and not “S SHUTDOWN,SUB=MSTR”
IF SUBMSTR=NO
  S SHUTDOWN.IPL,SUB=MSTR
  C &TASKNAME.&STCID
ENDIF
* make sure we were started with a good STCID and not the script library unit.
IF &STCID = 0303     ← x’0303’ is the unit for “SYS1.COMMANDS’
 STCID = IPL         ← Change it to “IPL”
ENDIF
…..
```

&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 "IF STRING" or "MSG 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
```

....
*
IF STRING @31 "UPDATE"

..... lots of other commands

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

.....

&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**"

.....

&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**"

.....

&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”

.....
```

&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 TESTJOB"

.....

&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”

.....

&SCRIPT or &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 AUTOIPL” ← Member name in the Commands PDS

.....

&SCRALL - Currently executing Script Dataset name and member STATIC variable:

&SCRDSN - 1 to 44 character Script dataset name and membername (if from a PDS) that is currently executing this command. If the script is run via "DD *", then the &SCRALL is "INLINE"

Example:

```
........
WTO The current Script Dataset is &SCRALL
**** Results in the following WTO:
"The current Script Dataset is Joe.SCRIPTS.DATA(TODAY)"

.....
```

&SCRDSN - Currently executing Script Dataset name STATIC variable:

&SCRDSN - 1 to 44 character Script dataset name that is currently executing this command. If the script is run via “DD *”, then the &SCRDSN is “INLINE”

Example:

....
WTO The current Script Dataset is &SCRDSN
**** Results in the following WTO:
“The current Script Dataset is Joe.SCRIPTS.DATA” ← normally the commands PDS

.....

&Wnn (a specific word of message being processed)

The &Wnn 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 use of the IFSTR, IFWORD, IF WORDS, MSG WORD or MSGWORDS 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".

Parameters:
NONE

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

```
........
MSGWORD 0 ABC1234I
WTO &W07 &W08 &W04 &W05 &W00
**** 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 "IF WORD" or "MSG 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 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

	00	01	02	03	04	05	06
Example: Message is:	**ABC12345**	**This**	**is**	**a**	**test**	**system**	**UPDATE**

....
*
MSGWORD 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"

.....

&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”

.....

&<variable> - use previously set STATIC user variable:

&<variable> - 1 to12 character variable (named within < > (greater than/ less than characters)) which will identify the 1 to 16 character actual variable which was set previously.

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”
.....
```

SETVAR - Setting Persistent Variables

<*Variable name*> 0123456789abcdef (up to 16 characters or spaces)
$$BLANKS$$ (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 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 VARABLE>, 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 to create a variable with the startup time for CICS001:

If you were to specify “SETVAR <MY_CICS001> Time=&HH:&MM”

SyzCMD/z will create a variable named “**MY_CICS001**”
which will contain: “**TIME=20:23**”

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) (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:

```
........
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)**
$$BLANKS$$ **(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 will cease to exist when the task that created the script 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 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 VARABLE>, 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 to create a variable with the startup time for CICS001:

If you were to specify "SETVAR <MY_CICS001> Time=&HH:&MM"

SyzCMD/z will create a variable named "**MY_CICS001**"
which will contain: "**TIME=20:23**"

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) (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:

```
........
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)**
$$BLANKS$$ **(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 VARABLE>, 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 to create a variable with the startup time for CICS001:

If you were to specify "REPVAR <MY_CICS001> Time=&HH:&MM"

SyzCMD/z will check for a variable named "**MY_CICS001**"
And if found will replace the contents with: "**TIME=20:23**"

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)**
	$$BLANKS$$	**(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.

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 VARABLE>, 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 to create a variable with the startup time for CICS001:

If you were to specify "REPVAR <MY_CICS001> Time=&HH:&MM"

SyzCMD/z will check for a variable named "**MY_CICS001**"
And if found will replace the contents with: "**TIME=20:23**"

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:

```
........
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 VARABLE>, 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 to specify "DELVART   <MY_&W01>"

SyzCMD/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) (sill 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) (sill exists)
VRC=8	Variable did not already Exist, not created
VRC=9	Command failure (user error, see messages)

#variables - DYNAMIC Variables

The *#variable* control command is technically not a command at all, but is the representation of a dynamically created user variable. These variables don't exist until they are created by the script, and when used later in the script the value they represent will be inserted in their place. Currently there can only be UP TO 50 dynamic variables that can exist at any one time. These DYNAMIC variables can be inserted within character strings and will be processed "on the fly" and replace the variable with the string characters represented by that variable. For instance, if you wanted to create a command that issued a WTO with the current time, you could set the current time in a WTO message and include that entire string as the dynamic variable, you could state:

When setting, unsetting or resetting Dynamic variables, you reference them via their #variable name, when using them as a display item you add a ampersand "&" to the front of the #var_name name to make &#var_name. It's a little confusing at first, but it's necessary because it's a completely separate part of SyzCMD/z that inserts dynamic variables into messages and script lines than the part that creates or updates the dynamic variables. As far as SyzCMD/z is concerned, when displaying things or using in scripts, dynamic variables and static variables are treated the same. The only reason that a pound sign "#" is necessary is because SyzCMD/z needed to have a way to tell dynamic variables apart form static ones and plain old text.

For instance if you wanted to have SyzCMD/z display the current time in a WTO message, you would code:

WTO Starting CICS at &HH:&MMam

and it was 6:15 in the morning, you would see 6:15am displayed, because SyzCMD/z "knows" that &HH is the hours static variable, and &MM is the minutes, and that characters "am" are just text.

But if you set a dynamic variable of &TIME to 6:15 because you wanted to tell the operator to start CICS at 6:15 in the morning, and coded:

WTO Please make sure to start CICS at &STARTTIMEam, it would look for a variable called "STARTTIMEam" which it would not find because you have set the dynamic variable of STARTTIME, not STARTTIMEam. We "could" have allow the use of:

WTO Please start CICS at &STARTTIME am (note the 'am' is separated by a space.

But doing that would have made it very easy to mess up, and in fact during the initial customer testing of the feature, that's exactly what happened, so we decided to denote the dynamic variables and the static ones differently. We discussed which one should have the "#" and which one the "&" and decided that there are already hundreds of thousands of scripts using the static variable with an ampersand, and to change them would cause the customer base to rebel.

So, the actual example of use is as follows, and is described in more detail in the SETDVAR | DVAR section later in this manual.

SETDVAR = #WTO1 "WTO=The current time is &HH:&MM:&SS" which will insert the entire string and also replace the three STATIC variables &HH, &MM and &SS with their current values so that the actual command that is process is "WTO = The current time is 11:23:55", (Assuming it was really 11:23:55). So, you could then have a script command that was simply:

&#WTO1

which would be replaced by the entire command:

WTO=The current time is 11:23:55

Obviously, you could simply issue the WTO without using the dynamic variable and it would probably be easier to use but that wouldn't make a very good example, would it?

Parameters:
NONE REQUIRED

SETDVAR | DVAR

```
#VariableName      string   (or “string containing spaces or special chars”)
```

The SETDVAR (or just DVAR) command is used to create (set), reset, or remove (set to blanks) a dynamic variable. The Dynamic Variables are different from Static variables, because they are set by the user and not by SyzCMD/z or the operating system automatically. They can be thought of as “user” variables, and are completely under the control of the person writing the script.

The variable itself can be anything that starts with a US pound sign “#” and can be any other characters or numbers and most special characters, up to a total length of 10 bytes. The name is automatically changed to upper case, so #a01 and #A01 are the exact same variable.

The String that you set the variable to can be any combination of letters and numbers, with the only restriction that if it contains imbedded blanks, it must be contained within Double-Quotation Marks, (i.e. “). The total length of the string cannot exceed 50 bytes, and if it does, it will be truncated to 50 bytes.

The entire command therefore is the command name “SETDVAR” followed by the variable name (which must start with a #), and then the string you want to set that variable name to.

example:

```
SETDVAR  #MYID = “&TASKNAME, &TASKID submitted by &OWNER”
```

For a job called PAYROLL running as JOB number JOB000012 and submitted by CA-7, this sets the Dynamic Variable #MYID to the string:

```
PAYROLL, JOB00012 submitted by CA-7
```

So, every time you want to use that string, you merely code &#MYID. The ampersand tells SyzCMD/z that you want to “USE” the Dynamic variable #MYID and not set or reset it.

So, coding the following:

```
WTOH The name of this task is &#MYID.
```

Would result in the following highlighted (non-scrolling) message being displayed on the operator console:

```
The name of this task is PAYROLL, JOB00012 submitted by CA-7
```

You can also use the Dynamic Variable when you send emails and text messages. Simply code &#MYID wherever you want to use it. For example:

```
Email
        To: BWTEXT
        From: JOB_Monitor@SyzyygInc.com
        Subject: &HH:&MM &#MYID
Sendmail
```

The above would sent a text to the cell phone identified with the nickname "BWTEXT" that says "11:23 PAYROLL, JOB00012 submited by CA-7", assuming of course that it was 11:23am. The &HH:&MM is replaced by the actual hour and minutes at that specific point in time that the script was executed.

Note: You can add a message to the text as well, but we have found that when sending text messages, that the SUBJECT is better to use for short messages. If you use the MSG: command to also add a message, most cell phones treat it like a very long text that starts with the Subject and tacks the MSG: part (for as long as you make it) as if it came as one long text string.

If you want to "unset" or delete a Dynamic Variable, you use the same SETDVAR command and set the string to "" (two double quotes in a row with no intervening spaces)

i.e. SETDVAR #MYID = ""

You can test if a Dynamic variable exists at all (i.e. is some to "something") that same way you check a persistent or non-persistent variable (the ones enclosed in "<" and ">").

IF #MYID ON (or IF #MYID OFF)

If a Dynamic Variable is set to "something", checking for ON will render to "true"

i.e.

```
If #MYID $ON$
        Do this       ← do this if it's set to something
ELSE
        Do that       ← do that if it's not set to anything (i.e. if unused or blank)
ENDIF
```

You can also use #variables in combination with other variables and even with other #varibles. For instance, Let's say you have a message that you processed with the console message processing "MSGWORDS" to find a message that has the words “given to CICS” in it, when that message is found, all of the words in that message are parsed into separate variable words. Which is actually:

```
 DFHSI1517  applid Control  is      being     given    to     CICS.
    &W0        &W1     &W2  &W3     &W4       &W5   &W6  &W7
```

And assuming our CICS applid is “PRODSYSP”

You could then use the SETDVAR command(s) to compare things.

```
SETDVAR #A = &W1 @ 0(4)     (so now #A is “PROD”)
SETDVAR #B = &W1 @4(4)       (so now #B is “SYSP”)
SETDVAR #C = &W7 @ 0(4)     ( this gets rid of the period after “CICS”)
```

You can now use the # variables to send a text message to someone that says that Production SYSP CICS is now up.

Via

```
EMAIL
  TO: 8007672244@test.att.net
  From: Automation@SyzygyInc.com
  Subject: &LDOW &HH:&MM Production #C #B is now active.
SENDMAIL
```

So the person gets a text :

“Tuesday 09:18 Production CICS SYSP is now active.”

&LDOW is the “long format” of the Day of the week
&HH = current hour of the day
&MM = current minute of the day

You could also compare these #varaibles

```
If #A = #B       (from above, #A= “PROD”, #B=”SYSP”)
   WTO they are equal
ELSE
  WTO they are not equal
ENDIF
```

Obviously #A is NOT equal to #B in this case, but it’s a demonstration of the capability and power of #variables.

6. Other script commands

ACTCONs

Console-name

The ACTCON control command is used to cause the command script to define and activate a new extended MCS console. This console will be used by subsequent console-related commands (IF string, IF word(s), MSGString MSGWORD(s)), to be the scan source for the messages that the commands are scanning. Care should be taken with the use of this command. When messages are NOT being actively processed (by one of the mentioned commands), they are stored in the memory of the SyzCMD/z script task to await processing. This storage of messages uses some region size of the task, so be aware that 1,000 messages can use 1,000x256 bytes of storage (approximately 256K). While that's not very much, you can easily forget that you started the console and end up with tens of thousands of messages in your buffer. This not only wastes storage, but will increase the overhead of processing those messages when the commands are actually processed, eventually the region could run out of memory. You should probably try to make sure that you activate the console judiciously. Don't forget to deactivate the console when you are done using it to scan for commands. You can activate and deactivate the console as many times as you wish in the same script.

Parameters:

Console-Name (Optional), default is the name of the TASK running the script)

Note: only one console should be used in each script, and no two scripts may use the same console name at the same time. Variables can be used in the console name, (i.e. parts of the time/date).

Example:

```
........
* Wait for CICS to start (up to 5 minutes)
* the message we want is: DFHSI1517  applid Control  is   being given to CICS.
*                         00         01     02       03   04    05    06 07
S CICS123          ←-start CICS
MAXWAIT 5m         ← Wait up to 5 minutes once we start looking
**Use &MM&SS in the console name to make sure it's unique
ACTCONS=CICS&MM&SS  ← name of this console (any unique name)
MSGWORD 0 DFHSI1517
**Send operator a sticky note that it's up
WTOH CICS region &W01 is now available!
DEACTIVATE CONSOLE.....
.......
```

ASKOPER

(timer-option) Answer-A,Answer-B,Question-Text?

{Countdown timer capable via MAXWAIT=9999s|m|h parm or (9999s|m|h)}
ASKOPER (9999h|m|s) A,B,Text of question

Note: If either option "A" or option "B" are enclosed in parentheses, then that enclosed option will be the default used should the timer be set and expire.
i.e: on a timer expiration ASKOPER (9999h|m|s) (A),B,Text of question?
Results in answer option "A" being automatically supplied

The ASKOPER control command is used to cause the command script to issue a WTOR which allows one of two possible replies (or defaults to YES and NO), and uses the Question-Text to formulate the WTOR text. The Operator can respond to the script with one of the two possible replies (Or YES or No). The response is saved in the &RESPONSE field and is available to via that field and can also be tested with the new "IF RESPONSE" control command.

Possible answers (A or B) are always taken as all UPPER CASE unless the response is enclosed in single quotes 'ReSpOnSe', even if issued in lower case. That's because IBM translates the text to upper case when the reply is issued from the console.

By Default, this request will wait forever for the reply to the WTOR that is created by this command, if however, there is a desire to have a "count-down timer" feature, you can accomplish that in one of two simple ways. The first (and not recommended way which we refer to as the "using a sledge hammer to drive a thumbtack" approach), you can at any time before the "ASKOPER" script command occurs, insert a MAXWAIT=9999s|m|H command (where the s=seconds, m=minutes, h=hours), and when the ASKOPER script command executes, not only will the standard WTOR message be displayed, but also a message telling the operator (or console person), that a timer has been started and what will occur (i.e. that answer "a" or "B" will be defaulted to), when that timer expires should they not answer. Setting MAXWAIT back to MAXWAIT=FOREVER (or NONE) will undo that setting. Please be careful when using "MAXWAIT=", because it will be in effect until you change it to something else, so other commands which can make use of "MAXWAIT=" will also be affected by that command.

(Suggested method) Should you want to affect ONLY this particular ASKOPER script command, you can (and we recommend this as the main way to accomplish the countdown feature), specify before the answer "A" option the amount of time that you wish set the countdown timer for.
For example, "**ASKOPER (23s) YES,NO,Do you want to stop?**" will generate the console WTOR asking the console operator if they want to stop, and also set the countdown timer for 23 seconds, after which, if they have not answered, one of

the answer option (i.e. answer “B” which is “NO”) will be taken as if that was the response. Should that happen, a message is generated to syslog to document that it was auto-responded to, and not performed by the operator. The advantage of this method is that tit only applied to this one single “ASKOPER” script command and not any other script commands. This feature has the ability to allow you to ask questions and if no one responds in a reasonable amount of time, you can insert logic that will do something else by default.

Also handy with “IF WTOR” and “ASKOPER” is the fact that the response provided to the WTOR is available at any time within the script (until overridden with a newer response) and can be used via the variable &RESPONSE. For instance, if the script issued a “IF WTOR WEEKLY,MONTHLY,What type of run is this?” script command, and the console operator were to respond with “MONTHLY” (that’s answer option “B”), then you can interrogate the &RESPONSE variable as many times as necessary and use it in building commands and messages (including email and Text messages) because &RESPONSE will contain MONTHLY.

Parameters:

(9999h|m|s)	(Optional, Countdown timer, seconds is the default)
Answer-A	(Optional , default is “YES”)
,	(comma is required after “A” option)
	(if coded in parentheses this will be the default on a timer pop)
Answer-B	(Optional, default is “NO)
	(if coded in parentheses this will be the default on a timer pop)
	(if no selection is made for default, Option “B” will be that default)
,	(comma is required)
Question-Text	(any text to formulate the WTOR)

Note: „Question will issue a WTOR of “Question” with default Yes or No Reponses
<u>Reponses to this request are always returned as all upper case unless enclosed in single quotes</u>.

Example:

```
........
* Ask if they have run Accounting backup (yet)
ASKOPER ,,Have you run the Accounting Backup job yet?
IF RESPONSE = YES
** if yes do this
        WTO Okay, we can get started now…
ENDIF
IF RESPONSE = NO
** They said “NO”
        WTO No problem, I will start that for you now.
        F A,ACTGBKUP
        Wait Until Started ACTGBKUP    ← wait for it to start
        Wait until stopped ACTGBKUP  ←it started, now wait for it to complete
        WTO please check that the ACTGBKUP job finished normally
```

```
          (this could have been handled with a handshake to SyzMPF/z checking
the return codes, but for simplicity we will not do that.)

ENDIF

* Timeout example of above.  Give them 5 minutes, if no answer option B is
*automatically taken.
ASKOPER (5m)  ,,Have you run the Accounting Backup job yet?
IF RESPONSE = YES
** if yes do this
          WTO Okay, we can get started now…
ENDIF
IF RESPONSE = NO
** They said "NO" or we defaulted here after 5 minutes with no answer
          WTO No problem, I will start that for you now.
          F A,ACTGBKUP
          Wait Until Started ACTGBKUP    □ wait for it to start
          Wait until stopped ACTGBKUP  □it started, now wait for it to complete
          WTO please check that the ACTGBKUP job finished normally
          (this could have been handled with a handshake to SyzMPF/z checking
the return codes, but for simplicity we will not do that.)

ENDIF

: Ask about the Weather

ASKOPER  Sunny,Cloudy, How's the weather outside?
** only "SUNNY or "CLOUDY" are allowed here.
IF RESPONSE = Sunny
** they said "SUNNY"
          Wto That's nice, I'm very happy for you
ENDIF
IF RESPONSE = CLOUDY
** they said "CLOUDY"
          WTO OMG!!!  Did you bring an umbrella today?
ENDIF
```

Note: on expiration of the timer set (2 minutes in the commands below)

ASKOPER (2M) YES,NO,Are the CICS files closed yet?
    Results in option **"NO"** being set after 2 minutes
ASKOPER (2m) (YES),NO,Are the CICS files closed yet?
    Resutls in option **"YES"** being set after 2 minutes
ASKOPER (2m) YES,(NO),Are the CICS files closed yet?
    Results in option **"NO"** being set after 2 minutes
ASKOPER (2m) (YES),(NO),Are the CICS files closed yet?
    Results in option **"NO"** being set after 2 minutes
.....

DEACTivate

Console

The DEACT control command is used to cause the command script to deactivate and remove the extended MCS console established via the ACTCON command (described previously). Care should be taken with the use of this command. When messages are NOT being actively processed (by one of the console message realated commands (IFSTR, IFWORD(s), MSGSTR MSGWORD(s)), they are stored in the memory of the SyzCMD/z script task to await processing. This storage of messages uses some region size of the task, so be aware that 1,000 messages can use 1,000x256 bytes of storage (approximately 256K). While that's not very much, you can easily forget that you started the console and end up with tens of thousands of messages in your buffer. This not only wastes storage, but will increase the overhead of processing those messages when the commands are actually processed, eventually the region could run out of memory. You should probably try to make sure that you activate the console judiciously and deactivate it when the messages are not deemed to be useful any more. Don't forget to deactivate the console when you are done using it to scan for commands. You can activate and deactivate the console as many times as you wish in the same script.

Parameters:

Console (Optional) Only one console can be used at a time within each script.

Note: only one console should be used in each script, and no two scripts may use the same console name at the same time. Variables can be used in the console name, (i.e. parts of the time/date).

Example:

```
........
* Wait for CICS to start (up to 5 minutes)
* the message we want is: DFHSI1517  applid Control  is   being given to CICS.
*                               00         01      02     03   04      05    06  07
S CICS123            ←-start CICS
MAXWAIT 5m           ← Wait up to 5 minutes once we start looking
**Use &MM&SS in the console name to make sure it's unique
ACTCONS=CICS&MM&SS   ← name of this console (any unique name)
MSGWORD 0 DFHSI1517
**Send operator a sticky note that it's up
WTOH CICS region &W01 is now available!
DEACTIVATE CONSOLE.....
…….
```

DEBUG

Yes | <u>No</u> **(no is the default)**

The "DEBUG" parameter is used to tell SyzCMD/z whether or not to display DEBUGing information on the console and in the LOG. This is normally only to be used at the direction of Syzygy personnel, but the site may use it if they are testing out features and wish to know detailed information on what is being performed for them within the script. The default is "NO". **<u>This script command can also be used as a GLOBAL (parmlib).</u>**

Parameters:

Yes | <u>No</u> (no is the default)

Example:

```
........
* Set Simulation Mode
SIMULATE=Y
DEBUG=Yes
S CICS002
*➔ results in :   "<SIMULATED> S CICS002" being sent to the Console.
*Set Simulation mode off
SIMULATE=N
.....
```

DELAY

**** As of Version 9.0, this script command has been replaced with the much more flexible PAUSE script command outlined later in this manual. DELAY, PAUSE and WAIT are now completely interchangeable for all subcommands.**

nnn (in seconds ONLY)

Description:

Provide a simple wait for a specific number of seconds. This command has been superseded by the more powerful PAUSE command, but is maintained for compatibility with previous releases of the product.

Parameters:

nnn

Up to a 3 digit number of seconds to delay execution of further commands. The maximum allowed value of seconds is 999 (16 minutes 39 seconds). If more time is desired, multiple DELAY commands may be used in sequence.

Sometimes a time delay is desired between two commands. This is accomplished by the DELAY control command as follows:

```
DELAY=nnn       where nnn is the number of seconds (max of 999) to wait
  or
DELAY nnn
```

This will cause the program to wait the specified number of seconds before processing the next command.

Example:

```
........
* Bring down CICS001
F CICS001, CEMT P SHUT
* wait 15 seconds
DELAY 15
* Bring down CICS002
F CICS002,CEMT P SHUT
```

.....

DOMALL

This command has no parameters

Description:

This command provides a way to remove all highlighted commands and messages issued by SyzCMD/z in the current script. Normally this would only be performed at the end of the script or during a complex script that might generate a lot of highlighted messages to the console. Operators tend to get really upset when you leave them no space to see responses from commands they enter.

Parameters:

none

Example:

```
........
WTOH TEST1
WTOH test2
…
* clear the highlighted commands (that we issued)
DOMALL
.....
```

ECHO

adding[
A | All
I | Ifs
W|Wait
G|Global
B |Bypassed
C|Commands }

subtracting{
NONE or OFF
NOI |NOIfs
NOG|NOGlobal
NOW|NOWait
NOB |NOBypassed
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
C|Commands
N | None or O|Off
NOI |NOIfs
NOG|NOGlobal
NOW|NOWait
NOB |NOBypassed
NOC|NOCommands

A or All - Causes SyzCMD/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 SyzCMD/z to ECHO the global scripting commands to the console (currently those are limted to ECHO and EOSM). This is the default setting along with C.

NOGlobal - is the opposite of Global.

I or Ifs - Causes SyzCMD/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 SyzCMD/z to ECHO the issued system commands to the console and syslog, This is the default setting along with G.

NOCommands - is the opposite of Commands.

W or Wait - Causes SyzCMD/z to ECHO Wait-Related logic to the console and syslog, (DELAY, PAUSE, WAIT UNTILxxxx, etc.)

NOWait - is the opposite of Wait.

B or Bypass - Causes SyzCMD/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.

Example:

```
........
ECHO=NONE     <— First turn off all echoing
ECHO=Commands    <— Then turn on only command echoing
ECHO=WAIT     <---- also turn on Wait related (pause commands)
........
```

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 preceeding 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 job failure (User 0017 abend).

Parameters:
NONE

Example:

```
........
* See if we are running on the TEST LPAR the SYSID would be “TEST”
IF SYSID PROD   (This could also have been “IF &SYSID = PROD”)
   Command
   Command
ELSE            (*Note: The SYSID was not “PROD” so do this processing)
    Command
ENDIF
.....
```

Script Command Echoing.

SyzCMD/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 2.0 of the SyzCMD/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 on separate command requests (one per line)

EMAIL and SMS/Text related Commands

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

SyzCMD/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 either SMTP directly or the SyzMail/z component (if it is licensed and installed). SyzMAIL/z is a more capable method of handling email and it allows the use of Nicknames and some other features not possible with SyzCMD/z directly. SyzMAIL/z will then be 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 SyzCMD/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:

```
.....(ENDJOB – last step of a JOB) member contents...
If TYPE 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)

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.

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:

```
.....(ENDJOB – last step of a JOB) member contents...
If TYPE 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
.....
```

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 and if SyzMAIL is installed/active) 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:

```
.....(ENDJOB – last step of a JOB) member contents...
If TYPE 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
.....
```

To:

Email Address or Nickname or &Variable

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 SYZCMDnn startup parameter member of the system parmlib concatenation. This can be set to a variable such as “To: &notify”, 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:

```
.....(ENDJOB – last step of a JOB) member contents...
If TYPE 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
.....
```

From:

Email Address or Nickname or &Variable

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 SYZCMDnn startup parameter member of the system parmlib concatenation. This can be set to a variable such as “From: &notify”, 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:

```
.....(ENDJOB – last step of a JOB) member contents...
If TYPE 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
   From: &Notify@thissite.com       ← from whoever gets notified normally
   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
.....
```

Cc:

Email Address or Nickname or &Variable

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 SYZCMDxx startup parameter member of the system parmlib concatenation. This can be set to a variable such as “Cc: &notify”, 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:

```
.....(ENDJOB – last step of a JOB) member contents...
If TYPE 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
.....
```

Bcc: Blind carbon copy

Email Address or Nickname or &Variable

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 SyzCMDxx startup parameter member of the system parmlib concatenation. This can be set to a variable such as “Bcc: &notify”, 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:

```
.....(ENDJOB – last step of a JOB) member contents...
If TYPE 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
   Bcc: &EXECCLASS         ← send copy to production job log
   SENDMAIL
 ENDIF
ENDIF
.....
```

ReplyTo:

Email Address or Nickname or &Variable

The “ReplyTo:” 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 will override the From: address so that any response to this email will go to the ReplyTo address instead of the standard From: address. This is useful for sites which do not accept email back into z/OS and want instead to redirect any responses to another place. This setting is not required if the site wishes to use the “default” setting supplied in the SyzCMDxx startup parameter member of the system parmlib concatenation. This can be set to a variable such as “ReplyTo: &notify”, which will send the email or SMS text message with a Reply-to setting of 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:

```
.....(ENDJOB – last step of a JOB) member contents...
If TYPE 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
   Replyto: Noone@thissite.com   ← In case they answer us back.
   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
.....
```

SUBJect: or Subj:

Any text (including &variables)

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: &LPAR from &TASKNAME (&TASKID)” (results in (for PRODA lpar) a subject in the email of: PRODA 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:

Example:

```
.....(ENDJOB – last step of a JOB) member contents...
If TYPE 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
     SubJect: Job not yet ended     ← overrides the MAXCC message above
     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
.....
```

Message: or Msg:

Any text (including &variables)

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 SyzCMD/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).

Parameters:

Any text or valid &variable

Example:

```
.....(ENDJOB – last step of a JOB) member contents...
If TYPE 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
.....
```

ATTACH

Any JES or standard dataset (including &variables)

The "ATTACH:" command is used to attach JES datasets or MVS datasets (up to 3) to the body of an email. Variables may be used, and if they are valid (i.e. they exist) then they will be resolved at execution time. If no ATTACH command is provided, then no attachments will be sent (unless they are set up by default in the SyzCMD/z parmlib member, and then only JES2 datasets will be sent by default) in the email or SMS/Text message

Parameters:

NOTE The ATTACH= command must **not be placed after the MSG: command because it will be treated as a literal message and not the dataset or data to be attached.

Attach=JESJCL - Attaches the JCL images as output from the converter

Attach=JESMSGLG - Attaches the JES message log (WTO's issued by the task)

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

Attach=JCLIN - Attaches the JCL as submitted or started. (JCL before JES processing)

Attach=DSN=dataset name (up to 3 DSN's are currently supported v8.5)
----may be any sequential dataset or PDS member
ATTACH=DSN=my.dataset.name or ATTACH=DSN=my.pdsname(member)

Example:

```
.....(ENDJOB – last step of a JOB) member contents...
If TYPE JOB              ← if this is a Batch job
 If MAXCC > 0000              ← If the job is other than MaxCC of zero
   ATTACH=JESMSGLG
   ATTACH=My.dataset.name
   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
.....
```

MAXMSG

nnnn

The “MAXMSG” control command is used to tell SyzCMD/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:

```
.....(ENDJOB – last step of a JOB) member contents...
If TYPE JOB            ← if this is a Batch job
  If MAXCC > 0000             ← If the job is other than MaxCC of zero
    MAXMSG=150         ← in case the job had a lot of steps, default was 50 lines
    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
.....
```

ENDIF

The ENDIF control command is used to end a list of conditional commands set by a preceding 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  (This could also have been “IF &SYSID = PROD”)
   Command
   Command
ELSE
    Command
ENDIF
.....
```

EOSM

[V | Verbose | A | All
Y | Yes
N | No]

The EOSM control command is used to control whether or not the End Of Script Statistic Messages are displayed and whether or not they are displayed in detail, or just “normal” mode.

Parameters:

V | Verbose | A | All
Y | Yes
N | No

V or Verbose or A or All - Causes SyzCMD/z to display all 15 end of processing statistic messages to the console.

Y or Yes - Causes SyzCMD/z to display only the “normal” end of processing Statistic messages to the console. These messages include Total Commands read from input (or EXEC parms) and total commands issued. This is the default setting.

N or No - Causes SyzCMD/z to not display any end of processing statistics messages to the console.

Example:

```
........
EOSM=Verbose
........
```

End of Script Messages

SyzCMD/z now supports the creation of a series of messages when it finishes the processing of a script that shows the statistics breakdown of the processed messages. This GLOBAL command can be entered anywhere within the script being processed or in the EXEC parm of the job.

EOSM (End Of Script Message) Descriptions:

Default messages are underlined.

nnnnnnnn Commands were read from the EXEC parms

The preceding message shows the count of commands which were executed via the programs EXECution parameters, i.e.
//RUN EXEC PGM=SYZCMDZ,PARM='D A,L;D T;'

nnnnnnnn Commands were read from standard input

The preceding message shows the count of commands which were executed via the programs normal input file, i.e. //IEFREDR DD * or DSN=....

nnnnnnnn Commands bypassed due to IF/ELSE processing

The preceding message shows the count of commands which were bypassed due to IF/ELSE processing, i.e. an IF or ELSE statement was processed negatively causing commands within the IF-Nest to be bypassed.

nnnnnnnn Input lines were comments or blanks

The preceding message shows the count of commands which were read from input but were determined to be either blank lines or comment lines i.e. the line started with an asterisk "*".

nnnnnnnn Statements were processed as IF based commands

The preceding message shows the count of commands which were processed due to IF commands. These were the commands htat were found to exist within the confines of nested IF's.

nnnnnnnn IF-type command blocks were skipped

The preceding message shows the count of commands which were skipped from within nested IF's that were "failed" or found to be not true, and thus were not processed.

nnnnnnnn Was the highest IF nest level reached

The preceding message shows the count of the highest Nested-IF level reached. There is a utility maximum of 8 levels per script.

nnnnnnnn WAIT-type (DELAY,PAUSE,UNTIL) commands processed

The preceding message shows the count of commands which were deemed to be commands that caused the script to wait for some period of time. Future versions of SyzCMD/z will display the total am0ount of time waited.

nnnnnnnn ENDIF commands were observed

The preceding message shows the count of commands which were ENDIF commands.

nnnnnnnn ELSE command blocks were applicable

The preceding message shows the count of commands which were ELSE commands.

nnnnnnnn System commands were issued

The preceding message shows the count of commands which were actually issued to the system console by the SyzCMD/z facility for this script.

nnnnnnnn ELSE command blocks were skipped

The preceding message shows the count of commands which were contained within ELSE sections of the Nest-If blocks that have been skipped because the IF statement that began the If-block was found to be true, thus the ELSE has to ne found not-true and the commands contained within it are not executed.

nnnnnnnn WTOs or WTORs were issued

The preceding message shows the count of commands which were WTO or WTOR messages.

nnnnnnnn Outstanding reply requests were successful

The preceding message shows the count of Reply to Operator messages that were successfully replied to.

nnnnnnnn Outstanding reply requests were NOT successful

The preceding message shows the count of Reply to Operator messages that were NOT successfully replied to, i.e. either the reply-ID did not exist or did not match the restrictions of the REPLY command as specified in the script.

EXIT

```
[=nn]
STOPCODE=nn
```

The EXIT and STOPCODE control commands are used to stop (or abort) the execution of the script in progress. EXIT can be used to EXIT alone (without any other parameters which results in Return Code zero being set) or can be supplied with the STEP return code you wish to be set. If you use STOPCODE, it must ALWAYS have a return code set as part of the control command. This return code can be tested in later steps of the JOB or TASK that the script processor is running under.

Parameters:

=nn

nn - required for STOPCODE, Optional for EXIT, sets the STEP Return code.

Example:

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

GOBACK

The GOBACK control command is used to, if within a ++INCLUDE member (subordinate script), to immediately GO BACK to the calling Script and treats the return just as if we reached End-Of-File on the ++INCLUDEed Subordinate script. If not within a ++INCLUDE subordinate script, then the GOBACK command is treated as if we reached normal End-Of-File on the command script, similar to an EXIT command but with no support for a return code

Parameters:

NONE

Example:

```
……..
* Sample ++INLUCDE with a GOBACK
..Normal script commands
*now execute a sub-script called testme from test.pds
++INCLUDE INCDSN=test.pds(testme)
…. Next commands are from (TESTME)
IFStarted CICS01
        F CICS01,CEmT I tas
        F CICS01,CSTT ALL
ELSE
        GOBACK  ←This sends us back to our base script right after the ++INCLUDE
ENDIF
IFAFTER 05:00
        Do something else
ENDIF
…..
```

GOTO

labelname
++RESET

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. SyzCMD/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
++RESET

labelname - required. The name of the label to branch to within the script.
++RESET - optional. Resets the LOOP detection code.

Example:

```
.........
Some misc commands logic
......
GOTO=CICS2 ---------------------------->------------------v
                                                          |
::CICS1                                                   |
IF STARTED CICS001                                        |
EXIT=4                                                    |
ELSE                                                      |
   Do some other commands                                 |
ENDIF                                                     |
EXIT                                                      |
::CICS2        <<<-----------------------------------------
IF STARTED CICS002
EXIT=4
ELSE
   Do some other commands
ENDIF
EXIT
.....
```

General “IF” command usage

The “IF’ based commands which follow this page are free-form in that the they can be specified as “IF command” or “IFcommand” and both will be handled identically. Further, all other parameters on each of these queries, can contain any number of spaces between the parameters, such that:

“IF   command   variable” is the same as “IFcommand variable”

Any nesting of the commands within the scripts are purely for site readability as it does not matter to SyzCMD/z where a command starts on the line. In fact, it is suggested that indenting “nested IF’s” be performed so that the site can better read and develop the scripts manually.

```
IF command variable
          Do this
          Do this too
          IF othercommand
                    Do this
          ENdIF                    ←-note upper/lower case non-sensitive
EndIF
```

Is easier to read and understand than:

```
IF command variable
Do this
Do this too
IF othercommand
Do this
ENDif
EndIF
```

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
…
::RETRY
IF &MN/&DD  &HH:&MM GT 02/23  10:00
*** do some stuff if we are past the time
ELSE
*** do something else
   PAUSE 1m
  GOTO Retry
ENDIF
…
```

** note, this would probably be better served via “WAIT until 10:00” but we currently don’t support a “WAIT until “DATE””.

IF <variable> Check a persistent (or non-persistent) SET Variable

ON ←- means it's set to "something"
OFF ← means it's current un-set to anything
=|EQ "Value" (up to 16 characters including other variables and blanks, if quotes are used, they are part of the variable)
^=|NE "Value" (up to 16 characters including other variables and blanks, if quotes are used, they are considered part of the actual variable)

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 OFF parameter 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.

NOTE: If you are checking the settings of **ON** or **OFF**, **DO NOT** use an equal sign "=" or not equal "^=" or the characters "E" or "NE" because it will make the compare for the characters **"OFF"** or "**ON**", and not the "values" of just being set or not set.

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

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 AFTER

HH:MM

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
   PAUSE 5s
   $TI1-5,ABCD
   $SI1-5
ELSE
   IF AFTER 17:00
     $PI
     PAUSE 5s
     $TI1-5,CDEF
     $SI1-5
   ENDIF
ENDIF
.....
```

IF BEFORE

HH:MM

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
   PAUSE 5s
   $TI1-5,ABCD
   $SI1-5
ELSE
   IF AFTER 17:00
     $PI
     PAUSE 5s
     $TI1-5,CDEF
     $SI1-5
   ENDIF
ENDIF
.....
```

IF CLASS

=|EQ|^=|NE|>|GT|<|LT|/|GE|\|LE
X (where 'x' is any valid class 0-9 or a-z)

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
IF CLASS = P
  Wto  Production JOB &TASKNAME is running
  .....  (other commands)
ELSE
  WTO &taskname was not a production job because it ran in class=&EXECCLASS
ENDIF
.....
```

IF CPUID

=|EQ|^=|NE|>|GT|<|LT|/|GE|\|LE
4 character CPUID

IFCPUID

The IF CPUID control command is used to cause the command script to test for the last 4 digits of the hardware serial number (CPUID) of the processor complex that is executing the script. 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 CPUID you have elected to test for is found to not be the current CPUID 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 CPUID 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:
4 character CPUID

Example:

```
........
* Make sure it's the PRODUCTION box
IF CPUID A123               ← is this the A123 CPU?
   $P PRT15
   $PI
   PAUSE 5s
   $TI1-5,ABCD
   $SI1-5
ELSE
* not PRODUCTION, so might be TEST
    IF CPUID=B887
      $PI
      PAUSE 5s
      $TI1-5,CDEF
      $SI1-5
    ENDIF
ENDIF
.....
```

IF DOW

=|EQ|^=|NE
MONday | TUEsday | WEDnsday | THUrsday | FRIday | SATurday | SUNday
Optional : -FIRst|-SECond|-THIrd|-FOUrth|-FIFth

IFDOW

The IF DOW control command is used to cause the command script to test for the current, specific Day Of the Week. You are required to identify the Day of the Week that you wish to test for and you may specify that it is equal nor not-equal to that day. If you do not specify equal or not-equal, EQ will be assumed.

You may not specify more than one day at a time on a IFDOW command line. For instance "IFDOW = MON | WED or FRIday", would be true if the day of the week were EITHER of Monday, Wednesday or Friday. If you use the vertical bar for "or" make sure that you remember that you must leave a blank space before and after the "|". SO "IF DOW = MON|TUE|WED| may not operate as you expect, you should instead code it as "IF DOW = mON | tue | Wed". (Mixed upper and lower case are not relevant to the command). You won't know exactly which specific day it is, but you said OR (or "|'), so you just wanted it to be one of them. One important hint, you can't specify the AND (or &) because you cannot be both Monday and Tuesday. You can indeed code it that way, but it will always end up FALSE. 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.

Optionally, you may specify an "occurrence" of the Day of Week to denote the first through fifth occurrence of that specific DOW within the current month. You can specify the first through fifth occurrence of that specific DOW. That specification MUST be preceded by a dash "-", or it will be ignored

NOTE: The optional OCCURRENCE is NOT the "actual" Week of the Month as specified by ISO (ISO specifies that the "FIRST" week of any month MUST contain a Thursday. We don't know why they chose that as a definition, but it's almost useless from a MAINFRAME JOB scheduling standpoint to count weeks in that way so we decided to count "occurrences of a day" within the month instead of the vague ISO "week"). So to put it another way, it's the OCCURRENCE of that particular DOW for this month, for instance FIRST SUNDAY, SECOND SUNDAY, etc. Because of the vagaries of the ISO specification for WEEK of MONTH, it was decided to go by the occurrence of the days instead of the week number itself since most sites will want a specific DOW (first, second, etc.) and not a specific week as outlined by the ISO specs.

Note: BE VERY careful when you use the NE or ^= parameter (not equal), especially when you specify multiple days or the optional -First thorough -Fifth because if ANY of the options are true it can really mess up your day (logically speaking).

Parameters:

MONday|TUEsday|WEDnsday|THUrsday|FRIday|SATurday|SUNday|ANY

Optional:

-FIRst|-SECond|-THIrd|-FOUrth|-FIFth

One of the Days Of the Week are required. You may abbreviate the day to 3 characters or more.

Example:

```
........
* Make sure it's Saturday
IF DOW SATURDAY
  $P PRT15
  $PI
  PAUSE 5s
  $TI1-5,ABCD
  $SI1-5
```

```
ENDIF
* Make sure it's Saturday
 IF DOW SUNDAY
    $PI
    PAUSE 5s
    $TI1-5,CDEF
    $SI1-5
ENDIF

* if it's Monday, Wednesday or Friday start the CICS cleanup task
IFDOW MON | WED | FRI
         S CCLEANER
ENDIF


* make sure it's the first SUNDAY of the month and do backups


IF DOW SUNDAY -FIRST
         WTO First Sunday, so bring down CICS and start backups
         PAUSE 10s
         F CICS0,CEMT P SHUT
         WAIT UNTIL STOPPED CICS0
         S BACKUPS
* Wait for the last BACKUP to start (currently BACKUP9)
         WAIT UNTIL STARTED BACKUP9
* Now that it started, lets wait for it to end
         WAIT UNTIL STOPPED BACKUP9
* Now restart CICS0
         S CICS0
.....
```

IF DSN

= (optional)
Data.set.name
CATaloged - is cataloged
CREated - was created *TODAY* ← since midnight
DASD|ONDasd - exists on DASD
TAPE|ONTape - exists on TAPE
REFerenced - Was Referenced (touched) *TODAY*
MIGrated - Is in Migration Status

IFDSN

The IF DSN control command is used to cause the command script to test for the Existence of a dataset, and additionally, whether or not that dataset has been Created or referenced today (as in the current date that the script is running). You can also check to see if the dataset is migrated or exists on DASD or on tape.

This is particularly handy in the case where you need ot know if a dataset has been referenced or created today, for later processing, within a batch job so that you can set a condition code for the step, which can be tested later in the batch JOB or task.

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 DSN you have elected to test for is found to NOT exist in the state you are testing for (i.e. NOT Referenced TODAY if you have selected “IF DSN – Dataset.name **REF**erenced”), 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:
Data.set.name - The 1 to 44 character DSN to test
Status to test for:

CATaloged	-	is cataloged
CREated	-	was created **TODAY**
DASD|**OND**asd	-	exists on DASD
TAPE|**ONT**ape	-	exists on TAPE
REFerenced	-	Was Referenced (touched) **TODAY**
MIGrated	-	Is in Migration Status

Example:

```
........
**** See if we have updated Prod.smf.daily today and make sure it's not
**** migrated (because it's big)
IF DSN = PROD.SMF.DAILY REF       ← was it referenced today?
*** Yes, so don't run the daily job
  EXIT STOPCODE=04           ← Exit with RC=4 because we ran already today
ELSE
**** no, so make sure it wasn't migrated or On tape
  IF DSN PROD.SMS.DAILY MIG
  **** Yes, it's migrated so have HSM restore it
    S HREST,DSN=PROD.SMS.DAILY
    Pause 2M  ← give HSM a chance to do it's thing
    IF DSN PROD.SMS.DAILY MIG              ← try it again
    **** Yes still migrated, make sure HSM is up and if so give it more time
      STARTED DFHSM          ← make sure it's up, if not MSG will be sent to oper.
       PAUSE  2M             ← HSM is up, give it 2 more minutes before we give up
    ELSE
    **** not migrated, we can start our job now
    ENDIF
  ELSE
   **** not migrated, we can start our job now
  ENDIF
  S SMFDAILY           ← if we got here, we need to start the daily job
ENDIF
```

IF EXPIRED

No parms are necessary

IFEXPIRED

The IF EXPIRED control command is used to cause the command script to test if the previous WAIT has expired due to the setting of the MAXWAIT parameter or the maximum time set on the actual command (if supported like "IF WTOR @3m …". If the MAXWAIT is left to default (which is FOREVER) or not individual countdown timer is set then IFEXPIRED will always be FALSE. The only time the IF nest will be performed is when the WAIT has actually expired, and not ended normally. 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 WAIT expiration switch is found to have NOT been set, 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 previous WAIT was successful and did NOT expire). Indentation of IF levels is not required, but makes the script more readable to a human.

****NOTE : If the EXPIRED flag was set, the "IF EXPIRED" command test will "unset" it. This means that if the EXPIRED flag is set, you can only check it once, and then it's reset to "off". If you don't want to check the flag, but you want to reset it off, you can issue:**

```
MAXWAIT=off      ← turns off the maxwait processing
MAXWAIT=forever  ←makes the maxwait parm "forever"
MAXWAIT=none     ← turns off the maxwait processing
```

Parameters:

None

Example:

```
........
* see if MVSRES is mounted, if not mount it and submit backups
V A123,ONLINE
MAXWAIT=1M               ← wait up to 1 minute
WAIT FOR ONLINE V=MVSRES
IF EXPIRED               ← still not online
```

```
  EMAIL       ← send email to tell tech support
      To: techsupport@thissite.com
      From: SyzCMDz@thissite.com
      Subject: A123 did not vary online
      MSG:  I tried to vary A123 online, but it didn’t seem to work
  SENDMAIL
ELSE          ← it came online
  F A,MVSBKUP     ← so start the backups
ENDIF
.....
```

IF FULLCPUid

=|EQ|^=|NE|>|GT|<|LT|/|GE|\|LE

The full 5 character CPUID is checked (this is rarely necessary and only then in the case where the last 4 digits of the CPUID are the same at a site).

IFFULLCPUid

The IF FULLCPUid control command is used to cause the command script to test for the full 5 digits of the hardware serial number (CPUID) of the processor complex that is executing the script. 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 CPUID you have elected to test for is found to not be the current CPUID 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 CPUID 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:

5 character Full CPUID

Example:

```
........
* Make sure it's the PRODUCTION box
IF FullCPUID 0A123              ← is this the 0A123 CPU?
  $P PRT15
  $PI
  PAUSE 5s
  $TI1-5,ABCD
  $SI1-5
ELSE
* not PRODUCTION, so might be TEST
   IF FULLCPUID=BB887
     $PI
     PAUSE 5s
     $TI1-5,CDEF
     $SI1-5
   ENDIF
ENDIF
.....
```

IF LPAR

```
                =|EQ|^=|NE|>|GT|<|LT|/|GE|\|LE
                LPARNAME  (of this LPAR)
IFLPAR
```

The IF LPAR control command is used to cause the command script to test for the 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 one to 8 character LPAR name (as specified on the HMC) is required.

Example:

```
........
* Make sure it's the PRODUCTION system
IF LPAR PROD  (could also be asked using &LPAR variable as "IF &LPAR = PROD")
  $P PRT15
  $PI
  PAUSE 5s
  $TI1-5,ABCD
  $SI1-5
ELSE
* not PRODUCTION, so might be TEST
    IF LPAR=TEST
      $PI
      PAUSE 5s
      $TI1-5,CDEF
      $SI1-5
    ENDIF
ENDIF
.....
```

IF LVRC

=|EQ|^=|NE|>|GT|<|LT|/|GE|\|LE
Returncode (Single digit between 0 and 9)

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)**
>=|GE	**Greater than or Equal to (between 0 and 9)**
,=|LE	**Less than or Equal to (between 0 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 <= 4
***  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) (sill exists)
VRC=8	Variable did not already Exist, not created
VRC=9	Command failure (user error, see messages)

IF MaxCC

=|EQ|^=|NE|>|GT|<|LT|/|GE|\|LE
nnnn (4 character Maximum Condition Code of this task to check for)

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, IEFC452I=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 = 0000    (can also specify as a variable as IF &maxCC = 0000)
   (EXIT)                  ← nothing special to do
ELSE
  IF MAXCC > S000          ← some type of ABEND S001-SEFF
  ..... (other commands)
  ENDIF
  IF Maxcc > U000          ← some type of user abend
    ..... (other commands)
  ENDIF
  IF maxcc > 0011          ← maxcc is greater than 12
    ..... (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

RESULT (ENDED, ABENDED, FLUSHED, ACTIVE)

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, 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

=|EQ|^=|NE|>|GT|<|LT|/|GE|\|LE
Program Name (1 to 8 characters)

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

PROCSTEP-name-with-the-highest-ConditionCode-thus-far

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 STEP02C
IF MAXCC > 0000
  IF MAXPSTEP = STEP02C
    (do some commands)
  ENDIF
ENDIF
.....
```

IF MAXSTEP

=|EQ|^=|NE|>|GT|<|LT|/|GE|\|LE
STEP-name-with-the-highest-ConditionCode-thus-far

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

=|EQ|^=|NE|>|GT|<|LT|/|GE|\|LE
X (where 'x' is any valid class 0-9 or a-z)

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  (could also use variable as "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

MESSAGE ID (up to 17 characters)

****Note. This only applies if the script was started by the provided MPF processing exit that was supplied with SyzCMD/z. Using the SyzMPF/z product is a much better option than this but if you don't have SyzMPF/z, then you can certainly use this as a very good second choice.***

IFMSGID

The IF MSGID control command is used to cause the command script to test for the specific MESSAGE ID of a message that caused this script to be processed. You are required to identify the MESSAGE ID only if the message is something other than the name of the script member being processed. That's because 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, 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 17 characters)

The one to 17 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

=|EQ|^=|NE|>|GT|<|LT|/|GE|\|LE
Task Name (up to 8)

***Note. This only applies if the script was started by the provided MPF processing exit that was supplied with SyzCMD/z. Using the SyzMPF/z product is a much better option than this but if you don't have SyzMPF/z, then you can certainly use this as a very good second choice.*

IFMSGTASK

The IF TASKNAME control command is used to cause the command script to test for the specific 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 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 MSGTASK CICS456
    S DB2CONB
    ENDIF
ENDIF
.....
```

IF MYNAME

=|EQ|^=|NE|>|GT|<|LT|/|GE|\|LE
jobname

The IF MYNAME control command is used to cause the command script to test for the name of the script command processing JOB or TASK that the script processor is executing under. You are required to identify the JOBNAME that you wish to test for. You may not specify more than one JOBNAME 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 JOBNAME you have elected to test for is found to not be the current JOBNAME, 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 JOBNAME 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:
jobname

The one to 8 character JOB name that this task is executing as.

Example:

```
........
* Make sure I'm the DAYTIME task.
IF MYNAME DAYTIME  (could also use variable "IF &TASKname = Daytime")
  $P PRT15
  $PI
  PAUSE 5s
  $TI1-5,ABCD
  $SI1-5
ENDIF
.....
```

IF MYOWNER

=|EQ|^=|NE|>|GT|<|LT|/|GE|\|LE
userid

The IF MYOWNER control command is used to cause the command script to test for the UserID of the person or task that started SyzCMD/z, which is the RACF (or ACF/2, etc.) UserID of the submitter of the task or job that the SyzCMD/z processor is executing under. You are required to identify the USERID that you wish to test for. You may not specify more than one USERID 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 USERID you have elected to test for is found to not be the current UserID of this task, 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

The one to 8 character UserID that this task is executing under.

Example:

```
........
* Make sure I'm running under CICS01's id to have proper authority .
IF MYOWNER CICS01
   F CICS01,CEMT P SHUT
ELSE
   WTOH=Hey Operator, you didn't submit me from CICS01
   EXIT=12    exit code 12
ENDIF
....
```

IF MYSTEP

```
                        STEPname
IFMYSTEP
```

The IF MYSTEP control command is used to cause the command script to test for the current JOB or Task STEP that is executing SyzCMD/z. You are required to identify the JCL “STEP name” that you wish to test for. You may not specify more than one step 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 name you have elected to test for is found to not be the current STEP of this task, 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 STEP 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:

STEPname

The one to 8 character STEPname that this task is currently executing.

Example:

```
........
* Make sure I’m running STEP123.
IF MYSTEP STEP123
  EXIT=0     exit code 0
ELSE
  EXIT=12    exit code 12
ENDIF
....
```

IF MYPSTEP

ProcSTEP name

IFMYPSTEP

The IF MYPSTEP control command is used to cause the command script to test for the current JOB or Task procedure step name that is executing SyzCMD/z. You are required to identify the procedure step name that you wish to test for. You may not specify more than one procedure step 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 PROCSTEP Name you have elected to test for is found to not be the current PROCSTEP of this task, 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 PROCSTEP 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:

procstepname

The one to 8 character ProcStepName that this task is currently executing.

Example:

```
……..
* Make sure I’m running Proc step P1.
IF MYPSTEP P1
  EXIT=0     exit code 0
ELSE
  EXIT=12    exit code 12
ENDIF
….
```

IF MYSUBMETHod

=|EQ|^=|NE|>|GT|<|LT|/|GE|\|LE
(valid methods include INTRDR | CONSOLE | READER | REMOTE* | OMVS)

IFMYSUBMETHod

The IF MYSUBMETH control command is used to cause the command script to test for the way this task entered the system. You are required to identify the Submit method name that you wish to test for (INTRDR, CONSOLE, etc.). You may not specify more than one Submit Method 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 of the task you have elected to test for is found to not be the current owner of this task, 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 Submit Method 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:

Submit Method name

The one to 8 character Submission method that this task used to enter the system.

Example:

```
........
* Make sure I came from the internal reader .
IF MYSUBMETH NE INTRDR
  F CICS01,CEMT P SHUT
ELSE
  WTOH=Hey Operator, I have to be submitted from the internal reader.
  EXIT=12    exit code 12
ENDIF
....
```

IF MYSUBSYStem

=|EQ|^=|NE|>|GT|<|LT|/|GE|\|LE
(valid compares are MSTR | JES2 | JES3 | JESx | APPC |
OMVS)
IFMYSUBSYStem

The IF MYSUBSYS control command is used to cause the command script to test for the Subsystem this script is running under. You are required to identify the SubSYSTEM name that you wish to test for (MSTR, JES2, JESA, JES3, Etc). You may not specify more than one SUBSYSTEM 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 SUBSYSTEM you have elected to test for is found to not be the current owner of this task, 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 SUBSYSTEM 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:

SUBSYSTEM name

The one to 4 character SUBSYSTEM that this task is executing under.

Example:

```
........
* Make sure I'm NOT running under MSTR to have proper authority .
IF MYSUBSYS NE MSTR
  F CICS01,CEMT P SHUT
ELSE
  WTOH=Hey Operator, you started me under the Master Scheduler not JES
  EXIT=12    exit code 12
ENDIF
....
```

IF NOTIFY

=|EQ|^=|NE|>|GT|<|LT|/|GE|\|LE
Name on the JOBCARD NOTIFY=name parameter

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 = payroll  (could also use variable "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=ccuu | 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.

This message is updated and redisplayed every 15 seconds.

Parameters:
A=ccuu
V=volser
One of either A= or V= is required. You may not specify both on the same control command.
A=ccuu - 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
   WAIT FOR ONLINE V=MVSRES
   F A,MVSBKUP
ELSE
* If already online, start the backup
   F A,MVSBKUP
ENDIF
.....
```

IF ONLINE

A=ccuu | 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.
This message is updated and redisplayed every 15 seconds.

Parameters:
A=ccuu
V=volser
One of either A= or V= is required. You may not specify both on the same control command.
A=ccuu - 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
  WAIT FOR ONLINE V=MVSRES
  F A,MVSBKUP
ENDIF
.....
```

IF OSMOD

=|EQ|^=|NE|>|GT|<|LT|/|GE|\|LE
MODLEVEL
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

=|EQ|^=|NE|>|GT|<|LT|/|GE|\|LE
RELEASE
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

```
                =|EQ|^=|NE|>|GT|<|LT|/|GE|\|LE
                VERSION
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 PGMRname

=|EQ|^=|NE|>|GT|<|LT|/|GE|\|LE
Programmername (Generics supported)

IFPGMRname

The IF PGMRname control command is used to cause the command script to test for the JOBCARD field “ProgrammerName” of the JOB or TASK that the script processor is executing under. You are required to identify the Programmer name that you wish to test for. You may not specify more than one Programmername 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 Programmer name you have elected to test for is found to not be the current Programmer 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 Programmer 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:
Programmer Name

The one to 20 character JOBCARD Programmer name that this task is executing as.

Example:

```
........
* Make sure I’m one of the Accounting JObs.
IF PGMR = Account*
  Wto Setting up the special accounting print initiators.
  $PI
  PAUSE 5s
  $TI1-5,ABCD
  $SI1-5
ENDIF
.....
```

IF RESPONSE

= | EQ | ^= | NE| > | GT | < | LT | / | GE | \ |LE **Text**

IFRESPONSE

The IF RESPONSE control command is used to cause the command script to compare the Test operand to the most current response received from either the IFWTOR or the ASKOPER command. Both of those commands will update the REPONSE variable (which is displayable via the &RESPONSE variable), and so care needs to be taken when using IFRESPONSE to be sure that you are getting the response from the MOST CURRENT ASKOPER or IFWOTR command ONLY. The variable will remain set until the next ASKOPER or IFWTOR, or until the end of the script.

This command begins an "IF NEST" of commands that will be done only IF the condition being tested for is a true statement (i.e a match). If the compare is found to be (false), 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

Parameters:

Text	-	Previous response text from the MOST recent ASKOPER or IFWTOR command

Example:

```
........
* Ask if they have run Accounting backup (yet)
ASKOPER ,,Have you run the Accounting Backup job yet?
IF RESPONSE = YES
** if yes do this
        WTO Okay, we can get started now…
ENDIF
IF RESPONSE = NO
** They said "NO"
        WTO No problem, I will start that for you now.
        F A,ACTGBKUP
        Wait Until Started ACTGBKUP    ← wait for it to start
        Wait until stopped ACTGBKUP  ←it started, now wait for it to complete
        WTO please check that the ACTGBKUP job finished normally
        (this could have been handled with a handshake to SyzMPF/z checking
the return codes, but for simplicity we will not do that.)

ENDIF
```

```
: Ask about the Weather

ASKOPER  Sunny,Cloudy, How’s the weather outside?
** only “SUNNY or “CLOUDY” are allowed here.
IF RESPONSE = Sunny
** they said “SUNNY”
        Wto That’s nice, I’m very happy for you
ENDIF

IF RESPONSE = CLOUDY
** they said “CLOUDY”
        WTO OMG!!!  Did you bring an umbrella today?
ENDIF
.....
```

IF RACFGRP

=|EQ|^=|NE|>|GT|<|LT|/|GE|\|LE
RACFGroup

IFRACFGRP

The IF RACFGRP 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 RACFGropup 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 RACFGRP payroll
   Wto  Payroll JOB &TASKNAME is running
ELSE
   WTO &taskname was not a payroll job because it’s RACF group is &RACFGRP
ENDIF
.....
```

IF REXXRC

= | > | <   nnnn (Return Code)

IFREXXRC

The IF REXXRC control command is used to check the execution of the previous (the last one executed) REXX command. The Return can be compared to the nnnn (1 to 4 character) return code issued from the REXX exec. The condition code can be compared to be EQUAL '=', GREATER THAN ">" or LESS THAN "<" the numeric value being checked. Leading zeros are not necessary and if supplied are removed, thus return code "0004", "004", "04", and "4" are all the same.
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 REXXRC you have elected to test for is found to not be the current RETURN CODE from the most current REXX execution, 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 does NOT match the requirements for the one you are testing for). Indentation of IF levels is not required, but makes the script more readable to a human.

Parameters:
"=", ">", or "<" and nnnn (numeric return code)

The one to 4 character return code is required.

Example:

```
........
*
REXX=CHEKSYS
IF REXXRC > 4              **  Was return code 5 or more?
  $P PRT15
  $PI
  PAUSE 5s
  $TI1-5,ABCD
  $SI1-5
ELSE
* Less than 5 (4 or less)
```

```
    $PI
    PAUSE 5s
    $TI1-5,CDEF
    $SI1-5
ENDIF
.....
```

IF STARTed|ACTive

taskname [9999s|m|h or HH:MM:SS]
(Generic taskname supported via % and *)

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. Optionally you may request that the JOB/TASK not only be running, but must have been running for AT LEAST a specific amount of time, specified in (up to 4 digits) in Seconds, Minutes or Hours. Alternatively you can use HH:MM:SS format (Hours:Minutes:Seconds). If the JOB/TASK is found to be not running (or not running for AT LEAST the requested amount of time), 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 or has NOT been active for AT LEAST the specified amount of time). Indentation of IF levels is not required, but makes the script more readable to a human. This script command is interruptable so that at any time while it is waiting for the task to start, you can modify the waiting task to SKIP or STOP. “F taskname,SKIP” will end the wait and treat the IFSTARTED as if it was successfully satisfied. “F taskname,STOP” will completely end the script.

Parameters:

taskname [HH:MM:SS]

Taskname - required - specifies the JOB or TASK that you wish to test for.
HH:MM:SS - optional - specifies the amount of time that the Taskname must have been active.

Example:

```
........
* See if CICS001 is already running
IF STARTED CICS001                              <----------------IF LEVEL 1 begin
* Shut it down if so
  F CICS001,CEMT P SHUT
  PAUSE 1M
*   See if it’s still running
    IF STARTED CICS001  90s (started for 90 seconds)<----------------IF LEVEL 2 begin
```

```
        F CICS001,CEMT P SHUT IMM
        PAUSE 1M
*       See if it is STILL running
        IF STARTED CICS001                     <----------------IF LEVEL 3 begin
           Cancel CICS001
*       And wait for it to stop completely
            WAIT FOR STOPPED 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 Started CICS*       (some cics is running, but which one???)
IF Started %%CICS   (TSCICS, ABCICS, any two chracters before CICS will fit)
IF Started CICS001 45s  (has CICS been started for at least 45 seconds?)
If StArTEd CICS001 01:00:00  (has CICS001 been up for at least an hour?)
IF started CICS001 1h    (same as above, started for at least 1 hour)
.....
```

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 STEPRESULT(Stepname) or (Stepname.Procstep)

RESULT (ENDED, ABENDED, FLUSHED, ACTIVE)

IFSTEPRESULT

The IF STEPRESULT (so far if the task has not ended yet) control command is used to cause the command script to test for the specific STEP of a tasks result (so far) of the task we are executing under. 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, flushed or ACTIVE)

Example:

```
........
* See if The job ended or abended
IF STEPRESULT(Step07) = ABENDED
   (do some commands)
ELSE
   IF STEPRESULT(STEP07) = ENDED
      (do some commands)
   ELSE
      WTO STEP07’s result is &STEPRESULT(STEP07)    ← &STEPRESULT is in beta for V9)
   ENDIF
ENDIF
.....
```

IF STOPped | INACTive

Taskname
(Generic taskname supported via % and *)

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. This wait for the task to stop is interruptable, and you can modiuify the task to end the wait, “F taskname,SKIP” will do that, you can also enter “F taskname,STOP” at any time to completely end the script.

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
  PAUSE 1M
*   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
       PAUSE 1M
```

```
*       See if it is STILL running
        IF STARTED CICS001                           <----------------IF LEVEL 3 begin
           Cancel CICS001
*       And wait for it to stop completely
            WAIT FOR STOPPED 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

(9999h|m|s timer-option) @nn |@all (String) | 'String' | "String"

{**Countdown timer capable via MAXWAIT=9999s|m|h parm or (9999s|m|h)**}

*****Note the internal SyzCMD/z console should be activated before this command is included in the script. Idealy, you would want the console to be started before the message fragment that you are looking for could have been issued, because we can't see massages that were issued before we have a console to read from. Logically, you would probably Activate a console, then do something via script commands, and then use the IF command here to parse those messages generated by your script or as a result of the script. (i.e. Activate MYCONS, then Start CICS0001, then check for the "Control is given to CICS" message, then do something because of it).***

Description:

The IFSTR command (which can be used as IFSTRING or IFGSTR), allows the script to monitor all console messages looking for a specific string to be included within the message. There are two possible location parameters, @nn which limits the beginning point of the string that is being monitored for (relative to zero being the first byte of the message), or @ALL, which means that the "string" may occur anywhere within the message. SyzCMD/z will continue to monitor the messages searching for that string until it is found, or until the MAXWAIT (see MAXWAIT command) time expires. When the message is found, all words of that message are mapped to &wnn variables. Additional console message related commands will begin their search starting with the message found by this command. The MSGSTR command must be preceded by a CONSOLE activation (ACTCON=) command (see ACTCON. . 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 is NOT found at the specified location (before the expiration of MAXTIME), 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.

****Note: IF the IFSTR(ing) command is separated into two separate words "IF" and "STR(ing)", you (currently) can only have a single space between the two words "IF" and "STR(ing)".**

By Default, this request will wait forever to find the string from the console messages, if however, there is a desire to have a "count-down timer" feature, you can accomplish that in one of two simple ways. The first (and not recommended way which we refer to as the "using a sledge hammer to drive a thumbtack" approach), you can at any time before the "IF STRing" script command occurs, insert a MAXWAIT=9999s|m|H command (where the s=seconds, m=minutes, h=hours), and when the IF STRing script command executes, not only will the standard WTOR message be displayed, but also a message telling the operator (or console person), that a timer has been started. If the timer should expire, the IF is deemed a "FALSE" respnse. Setting MAXWAIT back to MAXWAIT=FOREVER (or NONE) will undo that setting. Please be careful when using "MAXWAIT=", because it will be in effect until you change it to something else, so other commands which can make use of "MAXWAIT=" will also be affected by that command.

(Suggested method) Should you want to affect ONLY this particular IF STRing script command, you can (and we recommend this as the main way to accomplish the countdown feature), specify (as the first parameter after the "IF STRing", the amount of time that you wish set the countdown timer for.
For example, "IF STR (23s) @ALL 'CICS2821I' " will set the countdown timer for 23 seconds, after which, a message was NOT generated with the CICS2821I "somewhere" within the text of a(any) message, the "FALSE" condition will be set. Should that happen, a message is generated to syslog to document that the message was not located. The advantage of this method is that tit only applied to this one single "IF STRing" script command and not any other script commands. This feature has the ability to allow you to wait only a reasonable amount of time for that one message, you can insert logic that will do something else by default.

Parameters:

(9999h|m|s) default is seconds
@nn or @all
(string) or "String" or 'String'

nn or all - required - Up to a 2 digit or "all", that specifies the beginning character number of the string to be searched for. The offset is based on zero with the first position of the first word being @00.
(string) – The one to 60 character string to search console messages for. The word case much match exactly (upper/lower case).

Example:

```
........
* Wait for CICS to start (up to 5 minutes)
* the message we want is: DFHSI1517  applid Control is being given to CICS.
S CICS123          ←-start CICS
MAXWAIT 5m         ← Wait up to 5 minutes once we start
ACTCONS=CICSCON         ← name of this console (any unique name)
IFSTR @all 'CICS123 Control is being given to CICS.'
*(could also have used IFSTR (5m) @all 'CICS123 Control is given to CICS.'
**YES, Send operator a sticky note that it's up
        WTOH CICS region &W01 is now available!
ELSE
**No, send an email and tell the operator we have a problem
    EMAIL
        TO:Support
        CC: pagethem
        From: Operations@SyzygyInc.com
        Subject: &SYSID CICS region CICS123 not started correctly
        MSG:  CICS &W01 started over 5 minutes ago and is still not active
    SENDMAIL
    WTOH CICS region CICS123 is not responding in time.  Call Support
ENDIF
DEACTIVATE CONSOLE
.....
```

IF SUBMITMETHOD

=|EQ|^=|NE|>|GT|<|LT|/|GE|\|LE
Submit method source (INTRDR, RDRn, STCINRDR, etc.)

IFSUBMITMETHOD

The IF SUBMITmethod 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 SUBMITMETHOD = R3.RDR1
   Wto  The job came from remote 3
   .....  (other commands)
ELSE
   WTO &taskname was not submitted from Houston, submit
method=&SUBMITMETHOD
ENDIF
.....
```

IF SUBMSTR

IFSUBMSTR Yes|No

The IF SUBMSTR control command is used to cause the command script to test to see if the current Task that is executing SyzCMD/z is running under SUB=MSTR control (under the Master Scheduler) or under JES2 or JES3 (i.e. NOT under SUB=MSTR). You are required to provide the "Yes" or "No" on the command to denote that you are asking that SUB=MSTR is "YES" (that you are executing under SUB=MSTR control), or "No" (that you are NOT executing under SUB=MSTR, i.e. under JES control). You may not specify anything other than Yes or No (or Y or N). 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 you are testing for is found to not be the current execution mode of this task, 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 mode 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:

Yes | No

Either Yes or No, is required.

Example:

```
........
* Make sure I'm running under control of JES so that I can use SYSOUT.
IF SUBMSTR=YES
  EXIT=12     exit code 12
ELSE
  Do whatever we want to do if under JES.
ENDIF
....
```

IF SYSID

```
              =|EQ|^=|NE|>|GT|<|LT|/|GE|\|LE
              SYSTEMID
IFSYSID
```

The IF SYSID control command is used to cause the command script to test for the SYSTEM 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 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
  PAUSE 5s
  $TI1-5,ABCD
  $SI1-5
ELSE
* not PRODUCTION, so must be TEST
   $PI
   PAUSE 5s
   $TI1-5,CDEF
   $SI1-5
ENDIF
.....
```

IF SYSPLEX

=|EQ|^=|NE|>|GT|<|LT|/|GE|\|LE
SYSPLEXNAME

IFSYSPLEX

The IF SYSPLEX control command is used to cause the command script to test for the SYSPLEX 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 one to 8 character SYSPLEX name is required.

Example:

```
........
* Make sure it's the PRODUCTION sysplex
IF SYSPLEX PRODPLEX
   $P PRT15
   $PI
   PAUSE 5s
   $TI1-5,ABCD
   $SI1-5
ENDIF

.....
```

IF TYPE | MYTYPE

Task Type (JOB, STC, TSU, APP)

IFTYPE

The IF TYPE control command is used to cause the command script to test for the current Task type, (Batch JOBS are “JOB”, Started Tasks are “STC”, TSO Users are “TSU” and APPC tasks are “APP”), that is executing this SyzCMD/z script. You are required to identify the TYPE that you are testing for (JOB, TSU, STC, APP) that you wish to test for. You may not specify more than one TYPE 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 TYPE you have elected to test for is found to not be the current TASK TYPE of this task, 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 TYPE 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:

tasktype (JOB, STC, TSU, APP)

The 3 character TASK TYPE that this task is currently executing.

Example:

```
........
* Make sure I'm running as a Started Task.
IF TYPE=STC
  Do my special stuff
ELSE
  EXIT=12    exit code 12
ENDIF
....
```

IF WORD

(9999h|m|s timer) n WORD (generics allowed, multiple words can be used as long as they are separated by a vertical bar "|" and they are treated as "or")

{Countdown timer capable via MAXWAIT=9999s|m|h parm or (9999s|m|h)}

This command is the same as MSGWORD, except that it allows the IF/ELSE structure to the request.

*****Note the internal SyzCMD/z console should be activated before this command is included in the script. Ideally, you would want the console to be started before the message fragment that you are looking for could have been issued, because we can't see massages that were issued before we have a console to read from. Logically, you would probably Activate a console, then do something via script commands, and then use the IF command here to parse those messages generated by your script or as a result of the script. (i.e. Activate MYCONS, then Start CICS0001, then check for the word "Control" of Control is given to CICS message, then do something because of it).***

Description:

The IFWORD command, allows the script to monitor all console messages looking for a specific WORD at a specific WORD offset to be included within the message. The word offset is designated relative to zero, with zero being the first word of the message. The script may specify multiple interchangeable words by using the "OR" designation "|" between the words. SyzCMD/z will continue to monitor the messages searching for that word at that specific word offset until it is found, or until the MAXWAIT (see MAXWAIT command) time expires. When the message is found, all words of that message are mapped to &wnn variables. Additional console message related commands will begin their search starting with the message found by this command. The MSGWORD command must be preceded by a CONSOLE activation (ACTCON=) command (see ACTCON. 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 is NOT found at the specified location (before the expiration of MAXTIME), 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.

(Suggested method) Should you want to affect ONLY this particular IF WORD script command, you can (and we recommend this as the main way to accomplish the countdown feature), specify (as the first parameter after the "IF WORD", the amount of time that you wish set the countdown timer for.
For example, "IF WORD (23s) 4 CICS123 " will set the countdown timer for 23 seconds, after which, a message was NOT generated with the 4th word of the message equal to CICS123, the "FALSE" condition will be set. Should that happen, a message is generated to syslog to document that the message was not located. The advantage of this method is that tit only applied to this one single "IF WORD" script command and not any other script commands. This feature has the ability to allow you to wait only a reasonable amount of time for that one message, you can insert logic that will do something else by default.

****Note: IF the IFWORD(S) command is separated into two separate words "IF" and "WORD(S)", you (currently) can only have a single space between the two words "IF" and "WORD(S)".**

Parameters:

(9999h|m|s) seconds is the default
n ← the word number of the message (begins at 0 (zero))
WORD | WORD2 | WORD3 … (multiple choices for that word)

n - required - Up to a 2-digit number, that specifies the word offset (relative to zero) of the Message word to be searched for. The offset is based on zero with the first position of the first word being 0.

WORD – The one to 23-character WORD that the messages are checked for matching. The word case much match exactly (upper/lower case)

Example:

……..

```
* Wait for CICS to start (up to 5 minutes)
* the message we want is: DFHSI1517  applid Control  is   being given to CICS.
*                         00         01     02       03   04    05    06 07
S CICS123           ←-start CICS
MAXWAIT 5m          ← Wait up to 5 minutes once we start looking
ACTCONS=CICSCON          ← name of this console (any unique name)
IF WORD 0 DFHSI1517
*( could also have used "IF WORD (5m) 0 DFHSI1517"  )
**YES, Send operator a sticky note that it's up
        WTOH CICS region &W01 is now available!
ELSE
**No, send an email and tell the operator we have a problem
    EMAIL
        TO:Support
        CC: pagethem
        From: Operations@SyzygyInc.com
        Subject: &SYSID CICS region CICS123 not started correctly
        MSG:  CICS &W01 started over 5 minutes ago and is still not active
    SENDMAIL
    WTOH CICS region CICS123 is not responding in time.  Call Support
ENDIF
DEACTIVATE CONSOLE
.....
```

IF WORDS

(9999h|m|s timer) WORDs   (multiple words (up to 6) words can be used)
Upper/lower case sensitive

This command is the same as MSGWORDS, except that it allows the IF/ELSE structure to the request.
{Countdown timer capable via MAXWAIT=9999s|m|h parm or (9999s|m|h)}

****Note the internal SyzCMD/z console should be activated before this command is included in the script. Idealy, you would want the console to be started before the message fragment that you are looking for could have been issued, because we can't see massages that were issued before we have a console to read from. Logically, you would probably Activate a console, then do something via script commands, and then use the IF command here to parse those messages generated by your script or as a result of the script. (i.e. Activate MYCONS, then Start CICS0001, then check for the words of "Control is given to CICS" message (in any order i.e. "is given Control to CICS' is the same as the actual message), then do something because of it).***

Description:

The IFWORDS command, allows the script to monitor all console messages looking for a specific WORDS (up to 6 delimited by at least one blank space) that MUST occur somewhere in the message being scanned. The words can appear in ANY order and MUST appear within the first 50 words of the message. The script may specify up to 6 words, but if specified, they must ALL be found within the same message. SyzCMD/z will continue to monitor the messages searching for that combination of words until they are found, or until the MAXWAIT (see MAXWAIT command) time expires. When the message is found, all words of that message are mapped to &wnn variables. Additional console message related commands will begin their search starting with the message found by this command. The MSGWORDS command must be preceded by a CONSOLE activation (ACTCON=) command (see ACTCON. 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 is NOT found at the specified location (before the expiration of MAXTIME), 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.

(Suggested method) Should you want to affect ONLY this particular IF WORDS script command, you can (and we recommend this as the main way to accomplish the countdown feature), specify (as the first parameter after the "IF WORD", the amount of time that you wish set the countdown timer for.
For example, "IF WORDS (23s) Hello World " will set the countdown timer for 23 seconds, after which, a message was NOT generated with the words "Hello" and World" within the message, the "FALSE" condition will be set. Should that happen, a message is generated to syslog to document that the message was not located. The advantage of this method is that tit only applied to this one single "IF WORDS" script command and not any other script commands. This feature has the ability to allow you to wait only a reasonable amount of time for that one message, you can insert logic that will do something else by default.

****Note: IF the IFWORD(S) command is separated into two separate words "IF" and "WORD(S)", you (currently) can only have a single space between the two words "IF" and "WORD(S)".**

Parameters:

(9999h|m|s) (parentheses required, seconds is the default)

WORD WORD2 WORD3 WORD4 WORD5 WORD6

WORDn – The one to 23 character WORD that the messages are checked for matching. The word case much match exactly (upper/lower case)

Example:

```
........
* Wait for CICS to start (up to 5 minutes)
* the message we want is: DFHSI1517  applid Control  is   being given to CICS.
*                         00         01     02       03   04    05    06 07
S CICS123            <-start CICS
MAXWAIT 5m           <- Wait up to 5 minutes once we start looking
ACTCONS=CICSCON          <- name of this console (any unique name)
```

```
IFWORDS    Control    given    CICS.    DFHSI1517
*( could also have used “  IFWORDS  (5m) Control given CICS.  DFHSI1517  “
**YES, Send operator a sticky note that it’s up
        WTOH CICS region &W01 is now available!
ELSE
**No, send an email and tell the operator we have a problem
    EMAIL
        TO:Support
        CC: pagethem
        From: Operations@SyzygyInc.com
        Subject: &SYSID CICS region CICS123 not started correctly
        MSG:  CICS &W01 started over 5 minutes ago and is still not active
    SENDMAIL
    WTOH CICS region CICS123 is not responding in time.  Call Support
ENDIF
DEACTIVATE CONSOLE
.....
```

IF WORKLOAD

=|EQ|^=|NE|>|GT|<|LT|/|GE|\|LE
Workload Name this task is using

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:

WORKLOAD**Name**

Example:

```
........
* See if we are a Production Batch group job
IF WORKload = PRODBAT
   Wto  Productionl JOB &TASKNAME is running
   .....  (other commands)
ELSE
   WTO &taskname was not a production job because it has
WORKLOAD=&workload
ENDIF
.....
```

IF WTOR

Answer-A(true),Answer-B(false),Question-Text?

{Countdown timer capable via MAXWAIT=9999s|m|h parm or (9999s|m|h))
Ex: IF WTOR (9999h|m|s) A,B,Question-Text?

Note: If either option "A" or option "B" are enclosed in parentheses, then that enclosed option will be the default used should the timer be set and expire.
i.e: on a timer expiration IF WTOR (9999h|m|s) (A),B,Text of question?
Results in answer option "A" being automatically supplied and acted upon

IFWTOR

The IF WTOR control command is used to cause the command script to issue a WTOR which allows one of two possible replies (or defaults to YES and NO), and uses the Question-Text to formulate the WTOR text. The Operator can respond to the script with one of the two possible replies (Or YES or No). The IFWTOR then will utilize the ELSE statement of a normal IF/ELSE pairing to do either the "YES" path (the statements immediately following the IFWTOR up to the next ELSE statement if the operator responded with the "YES" response (which is the FIRST part of the statement before the first comma.). If the operator were to respond with the second operand (the "NON" operand) then the statements up to the next ELSE statement are skipped and the statements which follow the ELSE statement are performed instead. There is no "other" alternative, the operator must answer with the first option or the second one. This command begins an "IF NEST" of commands that will be done only IF the condition being tested for is a option 1 (true) statement. If the operator instead responds with the second (false) reply, 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. Possible answers (A or B) are always taken as all UPPER CASE even if issued in lower case. That's because IBM translates the text to upper case when the reply is issued from the console. We are working on allowing both IPPER/lower case responses.

By Default, this request will wait forever for the reply to the WTOR, if however, there is a desire to have a "count-down timer" feature, you can accomplish that in one of two simple ways. The first (and not recommended way which we refer to as the "using a sledge hammer to drive a thumbtack" approach), you can at any time before the "IF WTOR" script command occurs, insert a MAXWAIT=9999s|m|H command (where the s=seconds, m=minutes, h=hours), and when the IF WTOR script command executes, not only will the standard WTOR message be displayed, but also a message telling the operator (or console person), that a timer has been started and what will occur (i.e. that answer "B" will be defaulted to), when that timer expires should they not answer. Setting MAXWAIT back to MAXWAIT=FOREVER (or NONE) will undo that setting.

Please be careful when using "MAXWAIT=", because it will be in effect until you change it to something else, so other commands which can make use of "MAXWAIT=" will also be affected by that command.

Should you want to affect ONLY this particular IF WTOR script command, you can (and we recommend this as the main way to accomplish the countdown feature), specify before the answer "A" option the amount of time that you wish set the countdown timer for.
For example, "**IF WTOR (23s) YES,NO,Do you want to stop?**" will generate the console WTOR asking the console operator if they want to stop, and also set the countdown timer for 23 seconds, after which, if they have not answered, one of the answer option (i.e. answer "B" which is "NO") will be taken as if that was the response. Should that happen, a message is generated to syslog to document that it was auto-responded to, and not performed by the operator. The advantage of this method is that tit only applied to this one single "IF WTOR" script command and not any other script commands. This feature has the ability to allow you to ask questions and if no one responds in a reasonable amount of time, you can insert logic that will do something else by default.

Also handy with "IF WTOR" and "ASKOPER" is the fact that the response provided to the WTOR is available at any time within the script (until overridden with a newer response) and can be used via the variable &RESPONSE. For instance, if the script issued a "IF WTOR WEEKLY,MONTHLY,What type of run is this?" script command, and the console operator were to respond with "MONTHLY" (that's answer option "B"), then you can interrogate the &RESPONSE variable as many times as necessary and use it in building commands and messages (including email and Text messages) because &RESPONSE will contain MONTHLY.

** This is a new 9.0 feature. if you are interested in the feature that will allow a 3rd response (the timeout defaulted one), it is in development and scheduled to be available as an option in Version 9.2 of SyzCMD/z

Parameters:

Optional: **(9999s|m|h) (if s, m and h are all omitted, s (seconds) is the default)**

Required:

Answer-A	(Optional (true option), default is "YES") (if coded in parentheses this will be the default on a timer pop)
,	(comma is required)
Answer-B	(Optional, (false option) default is "NO) (if coded in parentheses this will be the default on a timer pop) (if no selection is made for default, Option "B" will be that default)
,	(comma is required)
Question-Text	(any text to formulate the WTOR)

Note: ,,Question will issue a WTOR of "Question" with default Yes or No Reponses

Reponses are always returned as all upper case.

Example:

```
........
* Ask if they have run Accounting backup (yet)
IF WTOR ,,Have you run the Accounting Backup job yet?
** if yes do this
        WTO Okay, we can get started now…
ELSE
** if they say "NO"
        WTO No problem, I will start that for you now.
        F A,ACTGBKUP
        Wait Until Started ACTGBKUP    ← wait for it to start
        Wait until stopped ACTGBKUP  ←it started, now wait for it to complete
        WTO please check that the ACTGBKUP job finished normally
        (this could have been handled with a handshake to SyzMPF/z checking
the return codes, but for simplicity we will not do that.)

ENDIF

: Ask about the Weather

IF WTOR  Sunny,Cloudy, How's the weather outside?
** they said "SUNNY"
        Wto That's nice, I'm very happy for you
ELSE
** they said "CLOUDY"
        WTO OMG!!!  Did you bring an umbrella today?
ENDIF
```

Countdown timer of 2mintes for the operator to respond

```
IF WTOR (2m) NO,YES,Would you  like me to start CICS?
** they said NO
  WTOH Okay, I guess no one really wanted to work today.
ELSE
**  they said YES or we timed out at 120 seconds (2 minutes)
  WTO  CICS will be started
  S CICS123
** start CICS123 and wait 35 seconds before we consider it "up"
**  this form of WAIT until started checks constantly and the specific
**   copy of CICS123 (whatever their taskID is) has to be continusly active for
**   the whole 35seconds  (if it abends and restarts the timer starts over
  Wait until started CICS123 35s
  S CICSOPEN    *open the special files
ENDIF
```

```
IF WTOR (2M)  YES,NO,Are the CICS files closed yet?
 *       Results in option “NO” being set after 2 minutes
  STUFF to do if “YES”
ELSE
  STUFF to do if “NO”
ENDIF
IF WTOR (2m)  (YES),NO,Are the CICS files closed yet?
 *        Resutls in option “YES” being set after 2 minutes
STUFF to do if “YES”
ELSE
  STUFF to do if “NO”
ENDIF

IF WTOR (2m)  YES,(NO),Are the CICS files closed yet?
 *        Results in option “NO” being set after 2 minutes
STUFF to do if “YES”
ELSE
  STUFF to do if “NO”
ENDIF

IF WTOR (2m)  (YES),(NO),Are the CICS files closed yet?
 *        Results in option “NO” being set after 2 minutes
STUFF to do if “YES”
ELSE
  STUFF to do if “NO”
ENDIF
.....
```

INTERVAL

nnnn[S|M|H]

Description:

SyzCMD/z has many sub-commands which will wait for something to happen. "WAIT until" and also the Started and stopped commands are good examples of this. **The default "interval" time of these waits is 15 seconds**. Many times, it is a waste of time to wait for 15 seconds for something that may occur in a second or two and it was discovered that the system resources involved in reducing the wait interval to even 1 second was trivial. You can, with this command set that interval to any value from 1 second up to 9999

Parameters:

nnn [S|M|H]

nnnn - required - Up to a 4 digit number of seconds (the default), minutes or hours to wait to allow syzCMD/z to re-check the environment. The maximum allowed value of digits is 9999. "S" is the default and sets the preceding number to be that many seconds. You can also specify "M" for minutes or H for hours. If not specified, the default is 15 seconds (the SyzCMD/z default)

Example:

```
........
* Bring down CICS001, but it comes down quick so set interval of STOPPED to 2
seconds
INTERVAL 2s
F CICS001, CEMT P SHUT
WAIT until STOPPED CICS001
* set inteval back to 15 seconds
INTERVAL 15s
.....
```

LOG

“anything” (including variables
(CLOSE)

The LOG command allows the script to write anything they wish to the LOGDSN dataset (which must have been previously defined within the JCL //LOGDSN DD, the PARMLIB LOGDSN=parameter or a previous LOGDSN=dsn command in this script. Any text can be used (up to 80 bytes per line), and any number of lines can be created. Please remember to close the log dataset when you are finished “LOG (Close)” command and member which will be used by the LOG= command (described later in this manual). If the running task specifies a //LOGDSN DD, it will be overridden by this specification. However, using this parameter will allow you to eliminate the //LOGDSN DD completely from your task JCL and your startup parameters. The log may be opened and closed to the same or different datasets any number of times.

Parameters:

“anything”	-	any text, including variables (80 bytes total per line)
(CLOSE)	-	Close the LOG dataset

Example:

```
……..
LOGDSN=&SYSID.&TASKNAME.D&MN&DD&YY.T&HH&MM&SS T(1,0) NEW
** Results in a new dataset called “PMVS.SHUTDOWN.D121216.T115523 (1 track)
                ** assuming SYSID is PMVS, This task is called SHUTDOWN, and the
                current time is 11:55:23 on 12/12/2016
LOG=Shutdown was begun for &SYSID on &MN/&DD/&YY at &HH:&MM&SS
** Results in “Shutdown was begun for PMVS on 12/12/16 at 11:55:23
……  rest of commands
LOG=&SYSID SHUTDOWN complete on &MN/&DD/20&YY at &HH:&MM:&SS
** results in “PMVS SHUTDOWN complete on 12/12/2016 at 12:04:18
LOG (Close)             ← closes the log dataset
…..
```

LOGDSN

data.set.name | data.set.name(member)
SYSOUT | SYSOUT(class)
C | T (pri,sec)
NEW | MOD |SHR

The LOGDSN command allows the site to specify a dataset (or dataset and member which will be used by the LOG= command (described later in this manual). If the running task specifies a //LOGDSN DD, it will be overridden by this specification and if a LOGDSN= parameter was specified in PARMLIB it will also be overridden by this command. However, using this parameter will allow you to eliminate the //LOGDSN DD completely from your task JCL and your startup paramters.

Parameters:

SYSOUT -	Write the log to JES SYSOUT
SYSOUT(class) -	Write the lof to JES SYSOUT class "class' (only 1)
C=pri,sec -	Allocate in Cylinders (pri = primary, sec=secondary space)
T=pri,sec -	Allocate in Tracks (pri = primary, sec=secondary space)
NEW | MOD | SHR	Disposition at open NEW will create new dataset MOD will append to the dataset SHR will replace the dataset contents (if not empty)
Defaults -	C=1,2 SHR
NOTE:	Tape data sets and migrated data sets are not supported.

Example:

```
……..
LOGDSN=&SYSID.&TASKNAME.D&MN&DD&YY.T&HH&MM&SS T=1,0 NEW
** Results in a new dataset called "PMVS.SHUTDOWN.D121216.T115523 (1 track)
                ** assuming SYSID is PMVS, this task is called SHUTDOWN, and the
                current time is 11:55:23 on 12/12/2016
LOG=Shutdown was begun for &SYSID on &MN/&DD/&YY at &HH:&MM&SS
** Results in "Shutdown was begun for PMVS on 12/12/16 at 11:55:23
……  rest of commands
LOG=&SYSID SHUTDOWN complete on &MN/&DD/20&YY at &HH:&MM:&SS
** results in "PMVS SHUTDOWN complete on 12/12/2016 at 12:04:18
LOG (Close)           ← closes the log dataset
…..
```

MAXMSG

Nnnn (default is 50 lines)

Description:

The “MAXMSG” parameter is used to tell SyzCMD/z how many lines the maximum email message will contain. If this number is exceeded, the recipient will receive a message that the email “may” have been truncated. The Default is 50 lines. The maximum number of lines is 9999. **<u>This script command can also be used as a GLOBAL (parmlib).</u>**

Parameters:

nnnn : Max of 9999 lines (default is 50 lines)

Example:

```
........
* Bring down CICS001, but it comes down quick so set interval of STOPPED to 2
seconds
INTERVAL 2s
F CICS001, CEMT P SHUT
WAIT until STOPPED CICS001
* set inteval back to 15 seconds
INTERVAL 15s
.....
```

MAXWAIT

nnnn H|M|S or Forever (default is forever)

Description:

The “MAXWAIT” parameter is used to tell SyzCMD/z how much time that any “WAIT” related command (i.e. “Wait until stopped *taskname*”) is allowed to wait before the script is to “assume” a failure of the wait and move to the next lines of the script. A failure on the wait because of this setting causes the EXPIRED flag to be set, and this can be tested via the “IF EXPIRED” or “EXPIRED” command which is discussed in other areas of this manual. The default is “FOREVER” **This script command can also be used as a GLOBAL (parmlib).**

Parameters:

nnnn H|M|S : Max of 9999
  H|Hours
  M|Minutes
  S|Seconds
FOREVER ← sets the value to “forever”
OFF ← turns off the maxwait processing
NO or NONE ← turns off the maxwait processing

Example:

```
........
* Bring down CICS001, but it comes down quick so set interval of STOPPED to 2
seconds
MAXWAIT 1m          ← wait only for 1 minute max
F CICS001, CEMT P SHUT
WAIT until STOPPED CICS001
* see if we ended or failed because it took more than 1 minute
IF EXPIRED          ← if we exceeded 1 minute
        F CICS001,CEMT P SHUT,IMMED      ← give it more juice
ELSE
**** nothing, we obviously worked in under the 1 minute
ENDIF
.....
```

MLWTO|MLWTOE

Anything (including &variables)

The MLWTO control command is used to issue Multi-Line (or single line if only one line is identified) informational messages to the operators' console, there are no restrictions on the contents, which are sent exactly as they are written. All subsequent MLWTO lines are combined to a single message until a MLWTOE (MLWTO End) command, or no more MLWTO messages are provided and some other command is provided which "breaks" the MLWTO sequence. No message is necessary on a MLWTO**E** command.

Parameters:
anything or nothing

Up to 72 characters are sent following the MLWTO control parameter.

Example:

```
........
MLWTO
MLWTO ******************************************
MLWTO *                                        *
MLWTO * This is a test                         *
MLWTO   Asterisks are not necessary
MLWTO *                                        *
MLWTO ******************************************
MLWTOE   <-end of MLWTO
.....
```

MSGDROP

Description:

The MSGDROP command is used to cause console processing logic to “drop” the message currently in use and move to the next match. This is only necessary on occasions where there may be multiple matches in the console message buffers and you want to move to the next one.

For instance, you may be searching through many similar messages via MSGWORD, MSGWORDS, MSGSTRING, IFWORD, or IFWORDS, and you want to test each of them individually. Normally, once you find a match, you would consider yourself done, and you can move on in the script, but if you wanted to look for the “next” match of that message you just found, you first need to tell SyzCMD/z to forget about the current message, otherwise it just keeps going back and getting the same message each time, because once a message is found, SyzCMD/z positions itself “ON” that message so that you can process it via other commands. To get SyzCMD/z to skip to the next occurrence of that message (if any), you use the MSGDROP command. SyzCMD/z will still remember all of the words from the previously found message, until you find a new one that matches. IF you don’t find a new one (a countdown timer has expired), all of the words that made up the originally found message are still set and you can use them later in the script.

Parameters:

NONE

Example:

```
........
* Wait for CICS to start (up to 5 minutes)
* the message we want is: DFHSI1517  applid Control  is   being given to CICS.
*                                00         01      02      03   04     05    06  07
S CICS123              □start CICS(s)
S CICS124
S CICS125
MAXWAIT 5m □ Wait up to 5 minutes once we start looking
ACTCONS=CICSCON        □ name of this console (any unique name)
MSGWORD 0 DFHSI1517
**Send operator a sticky note that it's up
WTOH CICS region &W01 is now available!  (for CICS123)
*Wait for the next one
```

```
MSGDROP  drop the data from the one we have
MSGWORD 0 DFHSI1517
**Send operator a sticky note that it's up
WTOH CICS region &W01 is now available!  (for CICS124)
*Wait for the next one
MSGDROP  drop the data from the one we have
MSGWORD 0 DFHSI1517
**Send operator a sticky note that it's up
WTOH CICS region &W01 is now available!  (for CICS125)
*done
DEACTIVATE CONSOLE
.....
```

MSGSTRing

(9999h|m|s timer) @nn |@all (String) | ‘String’ | “String”

{**Countdown timer capable via MAXWAIT=9999s|m|h parm or (9999s|m|h)**}

***Note the internal SyzCMD/z console should be activated before this command is included in the script. Idealy, you would want the console to be started before the message fragment that you are looking for could have been issued, because we can’t see massages that were issued before we have a console to read from. Logically, you would probably Activate a console, then do something via script commands, and then use the IF command here to parse those messages generated by your script or as a result of the script. (i.e. Activate MYCONS, then Start CICS0001, then check for the “Control is given to CICS” message, then do something because of it).*

Description:

The MSGSTR command (which can be used as MSGSTRING or MSGSTR), allows the script to monitor all console messages looking for a specific string to be included within the message. There are two possible location parameters, @nn which limits the beginning point of the string that is being monitored for (relative to zero being the first byte of the message), or @ALL, which means that the “string” may occur anywhere within the message. SyzCMD/z will continue to monitor the messages searching for that string until it is found, or until the MAXWAIT (see MAXWAIT command) time expires. When the message is found, all words of that message are mapped to &wnn variables. Additional console message related commands will begin their search starting with the message found by this command. The MSGSTR command must be preceded by a CONSOLE activation (ACTCON=) command (see ACTCON

(Suggested method) Should you want to affect ONLY this particular MSGSTRing script command, you can (and we recommend this as the main way to accomplish the countdown feature), specify (as the first parameter after the “MSGSTRing”, the amount of time that you wish set the countdown timer for.

For example, “MSGSTR (23s) @ALL ‘CICS2821I’ ” will set the countdown timer for 23 seconds, after which, a message was NOT generated with the CICS2821I “somewhere” within the text of a(any) message, the “FALSE” condition will be set and the EXPIRED flag is set (which can be checked via IF EXPIRED). Should that happen, a message is generated to syslog to document that the message was not located. The advantage of this method is that tit only applied to this one single “MSGSTRing” script command and not any other script commands. This feature has the ability to allow you to wait only a reasonable amount of time for that one message, you can insert logic that will do something else by default.

Parameters: (9999h|m|s) (seconds is the default)
@nn or @all
(string) or “String” or ‘String’

nn or all - required - Up to a 2 digit or “all”, that specifies the beginning character number of the string to be searched for. The offset is based on zero with the first position of the first word being @00.

(string) – The one to 60 character string to search console messages for. The word case much match exactly (upper/lower case).

Example:

```
........
* Wait for CICS to start (up to 5 minutes)
* the message we want is: DFHSI1517  applid Control is being given to CICS.
S CICS123              ←-start CICS
MAXWAIT 5m             ← Wait up to 5 minutes once we start
ACTCONS=CICSCON          ← name of this console (any unique name)
MSGSTR @all ‘CICS123 Control is being given to CICS.’
*( could have used MSGSTR (5m) @all ‘CICS123 Control is being given to CICS.’
**Send operator a sticky note that it’s up
WTOH CICS region &W01 is now available!
DEACTIVATE CONSOLE
.....
```

MSGWORD

(9999h|m|s countdown timer) n WORD

(generics allowed, multiple words can be used separated by vertical bars "|" and are treated as "or")

{Countdown timer capable via MAXWAIT=9999s|m|h parm or (9999s|m|h)}

*****Note the internal SyzCMD/z console should be activated before this command is included in the script. Ideally, you would want the console to be started before the message fragment that you are looking for could have been issued, because we can't see massages that were issued before we have a console to read from. Logically, you would probably Activate a console, then do something via script commands, and then use the IF command here to parse those messages generated by your script or as a result of the script. (i.e. Activate MYCONS, then Start CICS0001, then check for the word "Control" of "Control is given to CICS" message, then do something because of it).***

Description:

The MSGWORD command, allows the script to monitor all console messages looking for a specific WORD at a specific WORD offset to be included within the message. The word offset is designated relative to zero, with zero being the first word of the message. The script may specify multiple interchangeable words by using the "OR" designation "|" between the words. SyzCMD/z will continue to monitor the messages searching for that word at that specific word offset until it is found, or until the MAXWAIT (see MAXWAIT command) time expires. When the message is found, all words of that message are mapped to &wnn variables. Additional console message related commands will begin their search starting with the message found by this command. The MSGWORD command must be preceded by a CONSOLE activation (ACTCON=) command (see ACTCON

(Suggested method) Should you want to affect ONLY this particular MSGWORD script command, you can (and we recommend this as the main way to accomplish the countdown feature), specify (as the first parameter after the "MSGWORD", the amount of time that you wish set the countdown timer for.

For example, “MSGWORD (23s) 4 CICS001 ” will set the countdown timer for 23 seconds, after which, if a message was NOT generated with the 4th word “CICS001” , the “FALSE” condition will be set and the expired flag is set (which can be checked via IF EXPIRED). Should that happen, a message is generated to syslog to document that the message was not located. The advantage of this method is that tit only applied to this one single “MSGWORD” script command and not any other script commands. This feature has the ability to allow you to wait only a reasonable amount of time for that one message, you can insert logic that will do something else by default.

Parameters:	(9999h|m|}s)	Parentheses required (seconds are the default)
	n	
	WORD | WORD2 | WORD3 …	

n - required - Up to a 2-digit number, that specifies the word offset (relative to zero) of the Message word to be searched for. The offset is based on zero with the first position of the first word being 0.

WORD – The one to 23-character WORD that the messages are checked for matching. The word case much match exactly (upper/lower case)

Example:

```
........
* Wait for CICS to start (up to 5 minutes)
* the message we want is: DFHSI1517  applid Control  is   being given to CICS.
*                         00         01     02      03   04    05    06  07
S CICS123          ←-start CICS
MAXWAIT 5m         ← Wait up to 5 minutes once we start looking
ACTCONS=CICSCON       ← name of this console (any unique name)
MSGWORD (5m) 0 DFHSI1517
** could also have used “  MSGWORD (5m)  0 DFHSI1517     “
**Send operator a sticky note that it’s up
WTOH CICS region &W01 is now available!
DEACTIVATE CONSOLE
.....
```

MSGWORDS

(9999h|m|s countdown timer) WORDs (multiple words (up to 6) can be used)

{**Countdown timer capable via MAXWAIT=9999s|m|h parm or (9999s|m|h)**}

***Note the internal SyzCMD/z console should be activated before this command is included in the script. Ideally, you would want the console to be started before the message fragment that you are looking for could have been issued, because we can't see massages that were issued before we have a console to read from. Logically, you would probably Activate a console, then do something via script commands, and then use the IF command here to parse those messages generated by your script or as a result of the script. (i.e. Activate MYCONS, then Start CICS0001, then check for the words of "Control is given to CICS" message (in any order i.e. "is given Control to CICS' is the same as the actual message), then do something because of it).*

Description:

The MSGWORDS command, allows the script to monitor all console messages looking for a specific WORDS (up to 6 delimited by at least one blank space) that MUST occur somewhere in the message being scanned. The words can appear in ANY order and MUST appear within the first 50 words of the message. The script may specify up to 6 words, but if specified, they must ALL be found within the same message. SyzCMD/z will continue to monitor the messages searching for that combination of words until they are found, or until the MAXWAIT (see MAXWAIT command) time expires. When the message is found, all words of that message are mapped to &wnn variables. Additional console message related commands will begin their search starting with the message found by this command. The MSGWORDS command must be preceded by a CONSOLE activation (ACTCON=) command (see ACTCON

(Suggested method) Should you want to affect ONLY this particular MSGWORDS script command, you can (and we recommend this as the main way to accomplish the countdown feature), specify (as the first parameter after the "MSGWORDS", the amount of time that you wish set the countdown timer for. For example, "MSGWORDS (23s) AbC and 123 " will set the countdown timer for 23 seconds, after which, a message was NOT generated with the words "AbC", "and" & "123" within the message, the "FALSE" condition will be set and the expired flag is set (which can be checked via IF EXPIRED). Should that happen, a message is generated to syslog to document that the message was not located. The advantage of this method is that tit only applied to this one single

"MSGWORDS" script command and not any other script commands. This feature has the ability to allow you to wait only a reasonable amount of time for that one message, you can insert logic that will do something else by default.

Parameters: (9999h|m|s) Parentheses required (default is seconds)

WORD WORD2 WORD3 WORD4 WORD5 WORD6

WORDn –	The one to 23 character WORD that the messages are checked for matching. The word case much match exactly (upper/lower case)

Example:

```
........
* Wait for CICS to start (up to 5 minutes)
* the message we want is: DFHSI1517  applid Control  is   being given to CICS.
*                          00          01       02     03   04    05    06  07
S CICS123            ←-start CICS
MAXWAIT 5m           ← Wait up to 5 minutes once we start looking
ACTCONS=CICSCON         ← name of this console (any unique name)
MSGWORDS    Control    given    CICS.    DFHSI1517
* could have used  "MSGWORDS  (5m)  Control   given   CICS.  DFHSI1517  "
**Send operator a sticky note that it's up
WTOH CICS region &W01 is now available!
DEACTIVATE CONSOLE
.....
```

PAUSE

nnnn[S|M|H]
FOREVER
UNTIL HH:MM | H:MM | %%:MM
Subparms: HIGH|LOW|MSG|NOMSG|LONG
**See “WAIT” for additional functions of PAUSE command.
i.e. STARTED, STOPPED, ONLINE, OFFLINE, VTAM|NET, UNUSED
**DELAY, PAUSE and WAIT are now interchangeable.

Description:

Provide a simple wait for a specific number of seconds, Minutes, Hours or UNTIL a specific time of day (uses 24 hour “military time” format). You can pause using seconds/hours/minutes beyond midnight, but you cannot use the PAUSE UNTIL control parameter to span beyond 24:00

When a task is in the PAUSE state, the task is “interuptable” meaning that it can be modified with any of several options. “HOLD” or “WAIT” will freeze the pause timer until the task receives a “GO” command from the console. “SKIP” which will immediately end the pause and go on to the next script command. “REstart” will end the pause and restart it at the beginning, (i.e. 90 seconds into a PAUSE 5mintes, you can enter “F taskname,RESTART” and the pause will be restarted at 5 minutes. Were you to instead reply “WAIT” and then 30 seconds later (which is 120 seconds into the wait), reply “GO” it will continue the 5 minute wait (at the 3 minutes left mark) as if nothing happened to interrupt that wait.

Parameters:

nnnn [S|M|H] HIGH|LOW|MSG|NOMSG Default=HIGH +| INT
UNTIL HH:MM %%:MM HIGH|LOW|MSG|NOMSG Default=HIGH +| INT

nnnn - required - Up to a 4 digit number of seconds (the default), minutes or hours to pause execution of script commands. The maximum allowed value of seconds is

9999 (166 minutes). “S” (seconds) is the default and sets the preceding number to be that many seconds. You can also specify “M” for minutes or H for hours. The units (S|M|H) may be in upper or lower case

UNTIL HH:MM or H:MM or %%:MM - Optional - Any time of day can be specified up until midnight “24:00". The Time requested is based on the CURRENT DAY. You cannot request a time beyond the current midnight (24:00). If you request a time that has already happened, no pause will occur and the script continues with the next command. For instance, if it is currently 4am (4:00 or 04:00) and you specified “PAUSE UNTIL 3:30", since 03:30 (3am) has already occurred, there will be no pause performed.

If you specify %%:MM, then the script will pause until the “next” hour’s minute that matches the “MM”. For instance, if you were to code PAUSE UNTIL %%:MM and the current time was 7:01pm (19:01), then the script would wait until 20:01 (8:01pm) before continuing.

HIGH|LOW|MSG|NOMSG – designate how to display the informational message on the console, (or NOMSG to not display it at all). Sometimes it is convenient to not display the wait or to display the wait as a non-highlighted message, this parameter allows that to be possible. Care should be taken with NOMSG because there is no indication of what (or even if) the SyzCMD/z task is waiting for anything.

When a valid time is specified, a HIGHLIGHTED (the default intensity) console message will be issued to let the operator know that the task is waiting for a specific time to occur as follows:

PAUSE may be interrupted at intervals (set via the INTERVAL=Ns command) either via the F TASKNAME,SKIP|NEXT|GO or F TASKNAME,STOP|KILL command. SKIP, NEXT and GO will simply “skip” the pause time and move on to the next command, “STOP” will completely stop the script and force a RC=12 condition code (because the script was terminated) This parameter will allow long waits to be ended in case there is no longer a reason to pause.
Note: Using this parameter changes the default from a “wait” (using no CPU time) to a interval wait, so instead of a complete swapped out wait for the entire duration,

the task will come back "alive" at the interval set via the default (or previous) INTERVAL= command to check the console for commands from the operator. Currently the only commands that pertain to PAUSE are "STOP" which stops the task completely, and "SKIP" which causes the current PAUSE to be "skipped". Interval waits use a VERY slight bit extra CPU time, so if you don't need the ability to interact with the TASK, then use LONG or HARD to "standard" wait with no tests for changes to that wait.

```
CMD10PI  MYJOB WAITING UNTIL 13:21 as requested
```

When the time occurs, the message will be automatically DOM'ed, and processing will continue with the next script command

Example:

```
........
* Bring down CICS001
F CICS001, CEMT P SHUT
* wait 5 minutes but let operator intervene if they want to.
PAUSE 5M LOWlight    (LOWlight keeps the message from showing highlighted
* or we could allow interruptions to the wait
Pause 5M Low INT     (lowlighted message, but allows console interruption)
* Bring down CICS002
F CICS002,CEMT P SHUT
.....
```

REPLY

```
                JOBNAME MESSAGEID TEXT  nn
REPLY
```

The REPLY control command is used to provide replies to outstanding WTOR's without needing to know the "actual" replyid. You must know the JOBNAME or the MESSAGEID and/or the TEXT of the message to be replied to. Sometimes am outstanding message will not be available (yet) when this line is executed. Version 8.5 of SyzCMD/z ads the ability to "retry" the entire REPLY operation up to 99 times. The SyzCMD/z script will wait for the duration of the current INTERVAL (set by the INTERVAL= command) and then retry the REPLY operation. If the message has not become known by the time the script has exhausted the retries, the operation will fail and the script will continue without issuing the response just as pre-8.5 scripts operated. It is "suggested" that you set the interval to something reasonable before using the "nn" (retries) option, otherwise the script could be delayed for extremely long periods of time.

Parameters:

```
                JOBNAME|*
                                MESSAGEID|TEXT|*
                                                        REPLYTEXT
                                                                        nn
```

JOBNAME – Required, but can be generic "*" -
The full JOBNAME or "*" is required.
"*" denotes ANY jobname associated with the message that you wish to reply to. Specifying "*" will require that the MESSAGEID parameter be specified and the first JOB which has the matching messageid outstanding will be answered to.

MESSAGEID - Optional (unless JOBNAME was specified as "*"). Otherwise the Optional TEXT can be used or "*". If "text" contains any spaces, it should be

enclosed in single quotes, i.e. ‘Text to check for’. One of these entries for MESSAGEID is required to be used.
You may specify a message ID or a character string of text within the entire message If multiple messages could be outstanding for a job you may want to use this parameter to designate which outstanding message to reply to. Single quotes must be used if the string of text contains embedded blanks.

REPLYTXT - Required
This is the contents of the answer you wish to supply to the outstanding reply in question. Single quotes must be used if the string of text contains embedded blanks.

nn - Optional
This is the RETRY number that designates the number of times to retry this check for the outstanding reply. The max number is 99. The normal scan interval (set by INTERVAL= command) controls the wait period between retries.

Example:

```
……..
* Reply “QUIT” to WTOR for MYJOB, the WTOR must contain the string “WAITING
FOR”
REPLY   MYJOB    ‘WAITING FOR’    QUIT

* Reply “YES” to any message outstanding from job MYJOB (retry it 3 times)
                                   (Interval is 15 sec, so total wait is 45sec)
INTERVAL 15s                       ←- set interval to 15 seconds
REPLY MYJOB    *    YES    3

* Reply to any issuance of the ICK003D message with a “U” (message can be from
any jobname)
REPLY   *   ICK003D    U
…..
```

REXX

REXXNAME (REXX exec Name)

The REXX control command is used to execute a REXX exec. This exec will be executed in a normal TSO like environment. If the ID that requested the exec to be executed does not have a TSO segment, then it is the same as if the EXEC were executed in BATCH via the IRXJCL program. Use the IFREXXRC (identified earlier in this manual) command to check the return code from the execution of this exec.

Parameters:

1 to 8 character REXX exec name

The one to 8 character REXX exec name is required.

Example:

```
........
*
REXX=CHEKSYS
IF REXXRC > 4               **  Was return code 5 or more?
   $P PRT15
   $PI
   PAUSE 5s
   $TI1-5,ABCD
   $SI1-5
ELSE
* Less than 5 (4 or less)
    $PI
    PAUSE 5s
    $TI1-5,CDEF
    $SI1-5
ENDIF
.....
```

SETRC

[=nnn]

The SETRC control command is used to set or reset the condition code of the entire script. You can use this to set any code or reset a code that has already been set by the script do you errors or other coding variables. Any error that occurs AFTER the SETRC command will be able to override the one provided by SETRC if it is higher than the SETRC provided code. SETRC does not affect the stop errors or any messages that get generated by those issues. This return code can be tested in later steps of the JOB or TASK that the script processor is running under. SETRC codes will always override any other SETRC code whether it is higher or not.

Parameters:

=nn

Example:

```
........
* IF CICS001 is running then set RC = 99 but keep going
IF STARTED CICS001
SETRC=99
ELSE
  Do some other commands
ENDIF
.....
```

**The step return code from this script sill be 99 if the SETRC is performed and 00 if not.

SHOWSYSTEMVALUES

The SHOWSYSTEMVALUES control command is used to display z/OS system, LPAR and physical hardware values on the console. These can be used in debugging and in showing which system values can/should be used for the various IF-logic commands.

Parameters:

NONE

Example:

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

SHOWSYSTEMSVALUES
.....
```

SIMULATE

Yes|No (or anything but Yes will result in No)

The SIMULATE control command is used to cause SyzCMD/z to "simulate" the issuing of commands and replies to the system. The site can use this command to turn the Simulation Mode of SyzCMD/z on via "SIMUALTE=Yes" or off via "SIMULATE=No".

Parameters:

Yes | No

Example:

```
........
* Set Simulation Mode
SIMULATE=Y
S CICS002
*➔ results in :   "<SIMULATED> S CICS002" being sent to the Console.
*Set Simulation mode off
SIMULATE=N
.....
```

SYSMAIL =

Yes | <u>No</u>

The “SYZMAIL” control command is used to tell 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. If this is set to “NO” then email will be sent directly to SMTP by SyzCMD/z and Nicknames and other special SYZMAIL/z features will not be available.

Parameters:

Yes | <u>No</u> (NO is the default)

Example:

.....

SyzMail = Yes
…..

SUBMIT

Data.set.name | data.set.name(member)

The “SUBMIT” control command is used to submit a JOB or task to the internal reader. The job to be submitted can be a sequential dataset or a member of a PDS.

NOTE: A JCL DD card designating the Internal Reader to use for submission is necessary until Version 8.7 of SyzCMD/z. If you are at version 8.7 (or higher) you may disregard this NOTE.

Required JCL: //INTRDR DD SYSOUT=(A,INTRDR) <- “A” may be any valid class

Parameters:

DSN | DSN(member)

Example:

.....

Submit Sys1.JOBS(TESTME) ← submits the member TESTME from SYS1.JOBS

SUBMIT SEQ.DSN.JOB ← submits the dataset SEQ.DSN.JOB
…..

STCID

taskid

The “STCID” control command is used to allow alteration of the started task communications ID associated with a command script when started as a STC.

Previously if no “tasked” was assigned when starting a SyzCMD/z task as a started task (i.e. “S COMMANDZ”), the tasked assigned to the task would be the unit number that the COMMAND input dataset resided on. For instance, if the command script datasets “SYS1.COMMANDS” was located on Z1SYS1 (unit address x’ff20”), then the task would be COMMANDZ.FF20 and when dealing with the task (to modify it to issue commands or SKIP or STOP it), you would have to type “F FF20,SKIP”. This created a problem when you had multiple tasks all running with this same taskid. You could partially get around this IBM restriction by coding the TASKID in the start command (“S COMMANDZ.taskid”) so that you could reference the task (skip, stop, etc.) via “F taskid,SKIP”. But if you started several tasks with that same ID “S COMMANDZ,CICS1” and “S TALKTO.CICS1”, they would both use the same taksid (CICS1) and make it difficult to deal with them. Now you can alter the task internally within the script at any time during execution with the STCID command and you can display the tasked via the &STCID. Now you can have the script issue it’s own commands to itself (or make sure you have a unique ID via the new script commands.

Parameters:

Tasked - any characters (up to 8)

Example:

```
.....
EOSM=YES
ECHO=ALL
* it’s a shutdown for IPL so make sure we were started as a sub=mstr space
*  they may have just said “S SHUTDOWN” and not “S SHUTDOWN,SUB=MSTR”
IF SUBMSTR=NO
  S SHUTDOWN.IPL,SUB=MSTR
  C &TASKNAME.&STCID
ENDIF
* make sure we were started with a good STCID and not the script library unit.
IF &STCID = 0303     ← x’0303’ is the unit for “SYS1.COMMANDS’
 STCID  IPL          ← Change it to “IPL”
ENDIF
…..
```

STICKY

anything

The STICKY control command is used to issue highlighted informational messages to the operators console that will stay until cleared manually by the operator, 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. These messages can be removed ONLY via the UNSTICK command within a script. Otherwise they will stay on the console after the scritp ends, and the operator will need to use the "K E" command to clear the message manually. If you use UNSTICK, it only applies to the most current STICKY note created with the STICKY command. If you want a highlighted message that will be deleted when the script ends (or via a DOM or DOMALL command), please see the "WTOH" command.

Parameters:

anything or nothing

Up to 72 characters are sent following the STICKY control parameter.

Example:

....
STICKY Hey!, WAKE UP!!!!
.....

TimeZone or TZ =

Time zone offset to use (default = PST)

The “TimeZone or TZ” control command is used to tell 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)
.....
```

UNSTICK

ALL

Description:

This command provides a way to remove highlighted messages issued by SyzCMD/z in the current script via the STICKY command. Be careful that you don’t issue too many of the “STICKY” commands because when the consoles are in ROLL-DELETE mode as most of those consoles have 22 lines of text that can be displayed, and if you use them all up the operators tend to get very upset with you.

Parameters:

ALL – will unlhighlite and DOM all STICKY messages generated up to this time

Example:

```
........
WTOH TEST1          ← this goes away at the end of the script
Pouse 5s
….do something
STICKY test2        ← This stays even after the script is over
… do something else
* clear the highlighted commands (that we issued)
UNSTICK             ← remove the TEST2 sticky note
STICKY TEST3        ← This one will stay after the scritp ends
.....
```

When the above script runs, it will issue the “TEST1” highlighted message and wait 5 seconds, then do something else and then put another message up “TEST2” that looks just like the previous message, except that if the script ended there, TEST1 would go away and TEST2 would stay highlighted. The UNSTICK command would remove “TEST2” and then the next STICKY command would place the message “TEST3” on the console, which will stay after this script ends.

UNUSED

V=VOLSER or A=UnitAddress (one sub-parameter is required)

The UNUSED (and the "suggested" Wait Until Unused) control command is used to control execution of the next steps until a VOLSER or Unit Address is "unused", that is to say, is ONLINE but not allocated by any other address space in the system. The UNUSED command (and the longer "Wait Until UNUSED" or "WAITUNUSED" commands, will hold execution until the volume or address becomes free. If the device or volume is offline or not free, a Highlighted message is sent to the Operator Console and held on the console (assuming it's not in roll mode) to tell the operator that we are waiting for a device to free up. We check every 15 seconds to see if it has been free up yet, until it comes free. This is very useful in instances where you ABSOLUTELY cannot proceed until a device or volume is totally freed up by all users.

Parameters: (either V= or A= is required)

V=Volser
A=UnitAddress

V=Volser

Specify the volume serial number that you are waiting to be freed up. i.e. V=SYSRES

A=UnitAddress

Specify the hexadecimal Unit Address that you are waiting to be freed. i.e. A=1234

If you fail to specify either a VOLSER or UNITADDRESS, this command will fail and processing will continue with the next command in the script.

Example:

```
........
*Wait until the TSO volume TSO001 volume is free of users
UNUSED V=TSO001

* Wait until the TSO volume TSO001 (on X'3410' ) is free of all users
UNUSED A=3410

.....
```

VERBOSE

= **Yes | <u>No</u>** **(no is the default)**

The “VERBOSE” parameter is used to tell SyzCMD/z whether or not to display VERBOSE (longer) messages for most processes including WAIT messages. The VERBOSE messages normally show additional options that may be available to the console operator during a WAIT for some resource. The default is “NO”. **<u>This script command can also be used as a GLOBAL (parmlib) setting.</u>**

Parameters:

Yes | <u>No</u> (no is the default)

Example:

```
........
* Set Simulation Mode
SIMULATE=Y
VERBOSE=Yes
S CICS002
*➔ results in :   “<SIMULATED> S CICS002” being sent to the Console.
*Set Simulation mode off
SIMULATE=N
.....
```

WAIT UNTIL VTAM | NET ACTIVE

*NOTE ("ACTIVE" is not required, it's there for readability only)

**NOTE WAIT, PAUSE and DELAY are now fully interchangeable and all subcommands for any of them apply to all of them.

The WAIT UNTIL VTAM control command is used to cause the command script to test and optionally wait for the Z/OS Communications Manager (VTAM) to not only be started and running, but also for it to be able to function i.e. that the following message has been issued:

```
IST020I VTAM INITIALIZATION COMPLETE FOR CSVxRy
```

This is useful in instances (as during IPL) where is isn't enough to just know that VTAM has been started, but that it has reached a point in it's processing so that VTAM subsystems can be started without fear that they will attempt to initialize before VTAM is ready for them to be active. The command script processing will wait until VTAM is ready to begin normal processing, the command script will issue a console message and check back every "INTERVAL setting" number of seconds to see if VTAM is ready. A highlighted WTO is placed on the operator console to inform the operator that the command script is waiting for VTAM to be started (and ready for processing), as follows:

```
myjobname IS WAITING FOR VTAM TO START    Task modify available.
```

Where: myjobname is the name of the command script task

The operator may elect to SKIP this wait or STOP processing via:

```
F taskname,SKIP  (to skip this wait and proceed as if it was successful)  or
F TASKNAME,STOP (to stop waiting and end this script's processing)
```

Parameters:

None

Example:

```
........
* Make sure VTAM is ready for work
WAIT UNTIL VTAM      ACTIVE
.....
```

WAIT until STARTED

Taskname (9999s|m|h) 9999s|m|h

STARTED ***sometask***

nnnnS|M|H or [HH:MM:SS]

or

(nnnnH|M|S) nnnnH|M|S

MaxWait MinUpTime

**NOTE WAIT, PAUSE and DELAY are now fully interchangeable and all subcommands for any of them apply to all of them.

The WAIT UNTIL STARTED control command is used to cause the command script to test and optionally wait for a JOB or TASK to be running. Optionally you may request that the JOB/TASK not only be running, but must have been running for AT LEAST a specific amount of time, specified in seconds “9999s”, minutes “9999m”, hours “9999h” or HH:MM:SS format (Hours:Minutes:Seconds). If the JOB/TASK is found to be not running (or not running for AT LEAST the requested amount of time), the command script is paused and the system will be rechecked every “INTERVAL setting” number of seconds until the JOB/TASK comes active, (or is active for AT LEAST the amount of time specified). Also, you may set a MaxWaitTIme that the command will “wait” for the task to be started, if that time is set and expires before the task is started, the command is ended and the “EXPIRED” flag is set. That flag can be tested via the “IF EXPIRED” command (see IF EXPIRED in the table of contents). A highlighted WTO is placed on the operator console to inform the operator that a the command script is waiting for a JOB/TASK to be started, as follows:

```
myjobname IS WAITING FOR sometask TO START|BE ACTIVE FOR 9999S|M|H
                                                        Or      hh:mm:ss
If VERBOSE is requested :   CURRENTLY ACTIVE hh:mm:ss  Task modify available
```

Where: myjobname is the name of the command script task
sometask is the name of the JOB/TASK that we are waiting for
9999S|M|H the amount of time the task must be active to be true
hh:mm:ss is the amount of time that the JOB/TASK must be running

NOTE: If the 9999s|M|H or HH:MM:SS parm is not specified, it will not be reflected in the message)

The operator may elect to SKIP this wait or STOP processing via:

F myjobname,SKIP (to skip this wait and proceed as if it was successful) or

F myjobname,STOP (to stop waiting and end this script's processing)

Parameters:

taskname **(**9999H|M|S**)** 9999S\M\H or [HH:MM:SS]

Taskname - required - specifies the JOB or TASK that you wish to wait to become active
(9999s|M|H**)** – optional – Specified a MaximumWaitTime we should wait for the task to be started
9999s|M|H – optional – specifies the amount of time the task must be active after starting.
HH:MM:SS - optional but deprecated by 9999h|m|s- specifies the amount of time that the Taskname must be active.

Note as of version 9.0.4: Optionally a "maximum wait time" can be set either via the MAXWAIT= command, the MAXWAIT= startup variable or the internal MaxWait option on this line.

WAIT UNTIL STARTED **(9999H|M|S)** 9999H|M|S
MaxWait MaxUpTime

If the MAXWAIT time goes by, then the "IF EXPIRED" variable is set to TRUE|YES. The script will continue whether the MaximumWaitTime has transpired or if the task is found to be truly active, but if it is found active, then the IF EXPIRED process will be false. If MaximumWaitTime expires, then IF EXPIRED will be true.

There are (in this mode) two sets of times on the command line, the one in parentheses should normally go first, and it is the "maximum time" that the "**wait until started**" process will wait for the named task to start. It can be specified in (9999 Hours, Minutes or Seconds). This parameter is followed by the optional amount of time that the task must be actually in "continuous operation" to qualify. For instance, if the task in question is CICSA, and it must be up for at least a continuous 30 seconds to continue with the process, but you don't want to wait more than 5 minutes for it to be actually started for that 30 seconds, then you would code:

WAIT UNTIL STARTED CICSA **(5m)** 30s

Without the MaximumWaitTime parameter, the WAIT until STARTED taskname would wait forever for the task to be started. There are other ways ot handle this, (IF STARTED, etc.), but users have desired a Maximum time process for WAIT and this is to comply with that request.

If the process is as desired, i.e. CICSA is started and runs continuously for 30 seconds (not just started 4 times over the course of 30 seconds), then the wait is ended and the script continues with no settings changed.

If 5 minutes goes by, then the MAXWAIT of 5 minutes expires and the "IF EXPIRED" variable is set to TRUE or YES.

To utilize this in a script, a simple example is as follows, in this case we wait for CICS to be started for 30 continuous seconds, but something happened to CICA, so it did not start within the 5 minute "limit" that we set.

New enhanced example:

```
*Make sure CICS is up for at least 30 seconds, but only wait for a max of 5 minutes.
WAIT UNTIL STARTED CICSA (5m) 30s
IF EXPIRED
     •   Yes it is expired
*we have to do something about the fact that it's not running.
  WTOH CICSA did not start within the 5 minute time limit
  ...DO something about it
ELSE
     •   No, it started fine and ran for at least 30 secs
*nothing to do
  WTOH CICSA started okay
ENDIF
.......
```

Normal (pre 9.0.4) Example:

```
........
* Make sure TCP is up and ready for commands
WAIT UNTIL STARTED TCPIP

*Wait for CICS001 to be active AND make sure it's been active AT LEAST 5
minutes
WAIT UNTIL STARTED CICS001 5m
.....
```

WAIT until STOPPED

STOPPED **Taskname (9999s|m|h) 9999s|m|h**

taskname 9999s|m|h NORECHECK

(nnnnH|M|S) nnnnH|M|S
MaxWait MinUpTime

**NOTE WAIT, PAUSE and DELAY are now fully interchangeable and all subcommands for any of them apply to all of them.

The WAIT UNTIL STOPPED control command is used to cause the command script to wait for a JOB or TASK to NOT be running. As of version 9 of SyzCMD/z, there is an automatic delay of 1 seocnd after the task is found to be inactive. This is due to the possibility that on systems with multiple jobs in the queue with the same name, if one task ends on one processor and SyzCMD/z sees it it gone, and it were to stat up on another processor immeditly, it will be too late for SyzCMD/z to see the new task. The delay can be changed from the default 1s to (up to) 4 digits of seconds, minutes or hours. Even on very fast processors, 1second is normally sufficient. The command to globally change this delay is STOPCHECK=9999s|m|h, but it can be performed on a single basis simply be using the parm on this command as in "WAIT until stopped taskname 4s" will wait for the task to stop, then wait an additional 4 seconds and look again to see that it's still "gone". If the JOB/TASK is found to be running, the command script is paused and the system will be rechecked every "INTERVAL setting" number of seconds until the JOB/TASK ends or becomes inactive. Also, you may set a MaxWaitTIme that the command will "wait" for the task to be stopped, if that time is set and expires before the task stops, the command is ended and the "EXPIRED" flag is set. That flag can be tested via the "IF EXPIRED" command (see IF EXPIRED in the table of contents). A highlighted WTO is placed on the operator console to inform the operator that a the command script is waiting for a JOB/TASK to end, as follows:

SYZC102I myjobname IS WAITING FOR sometaskname TO COMPLETE. Task modify available.

Where: myjobname is the name of the command script task
sometaskname is the name of the JOB/TASK that we are waiting to end.

The operator may elect to SKIP this wait or STOP processing via:

```
F myjobname,SKIP  (to skip this wait and proceed as if it was successful)  or
F myjobname,STOP (to stop waiting and end this script's processing)
```

Parameters:

taskname (9999H|M|S) 9999S\M\H NORECHECK

sometaskname - required - specifies the JOB or TASK that you wish to wait to NOT be active

(9999s|m|h) – optional – Specified a MaximumWaitTime we should wait for the task to be stopped
9999 s|m|h - optional – specifies how long after the task is complete to wait before proceeding to the next command, this also makes sure that the task has not been restarted
NORECHECK- optional – Turns of the default habit of rechecking at the end of the timed wait (if specified) that the task is still down. It's possible that you might just want the task to stop and wait that much time, no matter what else may be started after the original stop.

Example:

```
........
* Make sure CICS001 is brought down
IF STARTED CICS001
    F CICS001, CEMT P SHUT IMM
    WAIT UNTIL  STOPPED CICS001     ← wait for CICS001 to stop
ENDIF
next command

* Make sure CICS001 is brought down and another doesn't just start right up
*  this waits for CICS0001 to stop, and be stopped for 5s, if 2 minutes goes by and
it still hasn't stopped, then set the EXPIRED flag and continue with the script.
IF STARTED CICS001
    F CICS001, CEMT P SHUT IMM
    WAIT UNTIL  STOPPED CICS001 (2m) 5s
    IF EXPIRED
         C CICS001
   ENDIF
ENDIF
next command

.....
```

WAIT until ONLINE | ONLINE

A=xxxx | V=volser 9999s|m|h

ONLINE

**NOTE WAIT, PAUSE and DELAY are now fully interchangeable and all subcommands for any of them apply to all of them.

The WAIT UNTIL ONLINE control command is used to cause the command script to test for and optionally wait for a UCB address or Disk or Tape volume serial to be mounted before proceeding. 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. If the UCB or VOLSER is found to be not mounted, the command script is paused and the system will be rechecked every "INTERVAL setting" number of seconds until the UCB or VOLSER is made online or mounted.

You can affect a maximum wait time for this command. This format is similar to how long to wait in other commands, but in this case it will cause the request to expire at that time, and it sets the "EXPIRED" switch, which can be checked via the "IF EXPIRED" command. Setting the time this way is the preferred method or adding a countdown timer (up to 9999 seconds, minutes or hours) to this "WAIT" command, or you can use the MAXWAIT= command (or the MAXWAIT parmlib member command) to set the maximum amount of time that you will allow for the volser or UnitAddress to come online. If the timer expires, the wait is ended and the script continues. The EXPIRED switch is set, so you can test if via the "IF EXPIRED" nested command. A highlighted WTO is placed on the operator console to inform the operator that a the command script is waiting for a UCB or VOLSER to be made online or mounted, as follows:

myjobname is waiting for volume volser to be mounted.

or

myjobname is waiting up to 9999s|m|h for Volume volser to be mounted.

or

myjobname is waiting for unit address uuuu to come online.

or

myjobname is waiting up to 9999s|m|h for Unit address uuuu to come online.

Where: myjobname is the name of the command script task volser is the name of the DISK or TAPE volume serial in question

xxxx is the hexadecimal UCB address in question

The operator may elect to SKIP this wait or STOP processing via:

F myjobname,SKIP (to skip this wait and proceed as if it was successful) or
F myjobname,STOP (to stop waiting and end this script's processing)

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:

```
........
* Make sure MVSRES is mounted
WAIT UNTIL ONLINE V=MVSRES

* Wait until Unix volume UCB is varied online
WAIT UNTIL ONLINE A=A100

*Wait up to 2 minutes for volser ZBHW00 to be mounted, if not mounted, then do
something

*vary ZBHW00 offline
* V 2300,offline
*wait for it to happen (forever if necessary)  no time specified assumes forever
WAIT until OFFLINE v=zbhw00
*Start the flashcopy
S FCPYZBHW     Flashcopy Brian's IPL volume
*wait for flashcopy to start
Wait until started FCPYZBHW
*wait for flashcopy to end
WAIT until stopped FCPYZBHW
*vary ZBHW00 online
V 2300,ONLINE
*wait till it happens (Flashcopy might hold it until background copy finishes)
```

```
WAIT until online v=ZBHW00 2m
If expired
*expired so we failed to have it mounted even after 2 minutes
   EMAIL
   TO:Brian_Westerman@SYzygyInc.com
   From: Mailbox@SyzygyInc.com
   Subject: &SYSID mount of ZBHW00 failed &HH:&MM:&SS on &MN/&DD/&YY
   MSG:  Recovery volume ZBHW00 did not come online after flashcopy
  SENDMAIL
ELSE
  WTO  proceed with the recovery..
ENDIF
.....
```

WAIT until OFFLINE | OFFLINE

```
                              A=xxxx | V=volser  9999s|m|h
    OFFLINE
```

**NOTE WAIT, PAUSE and DELAY are now fully interchangeable and all subcommands for any of them apply to all of them.

The WAIT UNTIL OFFLINE control command is used to cause the command script to test for and optionally wait for a UCB address to be varied offline or a Disk or Tape volume serial to be un-mounted before proceeding. 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. If the UCB or VOLSER is found to be online or still mounted, the command script is paused and the system will be rechecked every "INTERVAL setting" number of seconds until the UCB is varied offline or the VOLSER is un-mounted.
You can affect a maximum wait time for this command. This format is similar to how long to wait in other commands, but in this case it will cause the request to expire at that time, and it sets the "EXPIRED" switch, which can be checked via the "IF EXPIRED" command. Setting the time this way is the preferred method or adding a countdown timer (up to 9999 seconds, minutes or hours) to this "WAIT" command, or you can use the MAXWAIT= command (or the MAXWAIT parmlib member command) to set the maximum amount of time that you will allow for the volser or UnitAddress to go offline. If the timer expires, the wait is ended and the script continues. The EXPIRED switch is set, so you can test if via the "IF EXPIRED" nested command. A highlighted WTO is placed on the operator console to inform the operator that a the command script is waiting for a UCB to be varied offline or a VOLSER to be de-mounted, as follows:

myjobname is waiting for volume volser to go offline.

or

myjobname is waiting up to 9999s|m|h for volume volser to go offline.

Or

myjobname is waiting for unit address uuuu to go offline.

Or

Myjobname is waiting up to 9999s|m|h for unit address uuuu to go offline.

Where: myjobname is the name of the command script task volser is the name of the DISK or TAPE volume serial in question

xxxx is the hexadecimal UCB address in question

The operator may elect to SKIP this wait or STOP processing via:

```
F myjobname,SKIP  (to skip this wait and proceed as if it was successful)  or
F myjobname,STOP (to stop waiting and end this script's processing)
```

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:

```
........
* Make sure MVSRES is NOT mounted
WAIT FOR OFFLINE V=MVSRES

* Wait until Unix volume UCB is NOT online
WAIT FOR OFFLINE A=A100

*vary ZBHW00 offline
* V 2300,offline
*wait for it to happen (forever if necessary)  no time specified assumes forever
WAIT until OFFLINE v=zbhw00
*Start the flashcopy
S FCPYZBHW     Flashcopy Brian's IPL volume
*wait for flashcopy to start
Wait until started FCPYZBHW
*wait for flashcopy to end
WAIT until stopped FCPYZBHW
*vary ZBHW00 online
V 2300,ONLINE
*wait till it happens (Flashcopy might hold it until background copy finishes)
WAIT until online v=ZBHW00 2m
If expired
*expired so we failed to have it mounted even after 2 minutes
```

```
  EMAIL
  TO:Brian_Westerman@SYzygyInc.com
  From: Mailbox@SyzygyInc.com
  Subject: &SYSID mount of ZBHW00 failed &HH:&MM:&SS on &MN/&DD/&YY
  MSG:  Recovery volume ZBHW00 did not come online after flashcopy
 SENDMAIL
ELSE
 WTO  proceed with the recovery..
ENDIF
.....
```

WAIT Until UNUSED

V=VOLSER or A=UnitAddress (one sub-parameter is required)
9999s|m|h

**NOTE WAIT, PAUSE and DELAY are now fully interchangeable and all subcommands for any of them apply to all of them.

The "Wait until unused" control command is used to control execution of the next steps until a VOLSER or Unit Address is "unused", that is to say, is ONLINE but not allocated by any other address space in the system. The UNUSED or "Wait until Unused" commands, will hold execution until the volume or address becomes free. If the device or volume is offline or not free, a Highlighted message is sent to the Operator Console and held on the console (assuming it's not in roll mode) to tell the operator that we are waiting for a device to free up. We check every "INTERVAL setting" number of seconds to see if it has been free up yet, until it comes free. This is very useful in instances where you ABSOLUTELY cannot proceed until a device or volume is totally freed up by all users. You can affect a maximum wait time for this command by using the preferred method or adding a countdown timer (up to 9999 seconds, minutes or hours) to this "WAIT" command, or you can use the MAXWAIT= command (or the MAXWAIT parmlib member command) to set the maximum amount of time that you will allow for the volser or UnitAddress to go offline. If the timer expires, the wait is ended and the script continues. The EXPIRED switch is set, so you can test if via the "IF EXPIRED" nested command.

Parameters: (either V= or A= is required)

V=Volser
A=UnitAddress

V=Volser
Specify the volume serial number that you are waiting to be freed up. i.e. V=SYSRES

A=UnitAddress
Specify the hexadecimal Unit Address that you are waiting to be freed. i.e. A=1234

If you fail to specify either a VOLSER or UNITADDRESS, this command will fail and processing will continue with the next command in the script.

The operator may elect to SKIP this wait or STOP processing via:

```
F myjobname,SKIP  (to skip this wait and proceed as if it was successful)  or
F myjobname,STOP (to stop waiting and end this script’s processing)
```

Example:

```
........
*Wait until the TSO volume TSO001 volume is free of users
WAIT UNTIL UNUSED V=TSO001

* Wait until the TSO volume TSO001 (on X’3410' ) is free of all users
wait until unused A=3410


*wait for 2 minutes for a volume to come onlne or to be unused
* you may need this when you want to start a backup or flashcopy and need
* to make sure no one is using that volume

WAIT UNTIL unused V=ZBHW00 2m
If expired
*we didn’t get it online or it was and it was online but was being used
* so send a highlighted WTO to the operator
  WTOH  FAILURE to get the volume to ourselves
ELSE
*it worked
  WTO continue on with your work, the volume is clear
ENDIF
.....
```

WTOPREFix

= Yes | No (yes is the default)

The “WTOPREFix” GLOBAL setting parameter is used to tell SyzCMD/z whether or not to display the “* taskname *” in front of WTO command text and “*!taskname!*” in front of WTOR command text. The default is “YES”. This script command can also be used as a GLOBAL (parmlib) setting.

Parameters:

Yes | No (yes is the default)

Example:

```
........
* Turn off prefix (starts in ON position), assume taskname is MYTASK
WTO This is a test….
WTOPREF=NO
WTO This is a test too!
* results in :   * MYTASK * - This is a test….
* results in :  This is a test too!
.....
```

WTO

```
anything
```

The WTO control command is used to issue informational messages to the operator's 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

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. These messages can be removed at any time with the DOM or with the DOMALL (removes all WTOH's) command and they "ALWAYS" are deleted at the end of the script. I you want them to stay after the script is done, please see the "STICKY" command.

Parameters:

anything or nothing

Up to 72 characters are sent following the WTOH control parameter.

Example:

```
........
WTOH Hey!, WAKE UP!!!!
.....
```

WTOR

anything

The WTOR control command is used to issue a WTOR to the operator, a response from the operator is required and can be one of 3 responses, which are tested within the program, and execution of the script will pause until the operator replies to the outstanding message:
1) Y or YES - The logic of the script will continue uninterrupted
2) N or NO - Processing of the script will end
3) R - will redisplay the message

The WTOR is sent to the operator 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 and are followed by the standard:

Reply “Y” YES to continue, “N” NO to exit, or “R” to reshow the request.

Parameters:
anything or nothing

Up to 72 characters are sent following the WTOR control parameter.

Example:

```
........
WTOR Are you sure that the AS/400 is turned on?
.....
```

Results in the following on the console

```
     Are you sure that the AS/400 is turned on?
    *04 Reply “Y” YES to continue, “N” NO to exit, or “R” to reshow the request.
```

The operator must reply to the 04 attention message to allow the script to continue.

Cross references to other products

SyzCMD/z can operate in conjunction with the SyzAUTO/z product. SyzAUTO/z can schedule any SyzCMD/z command script at any time of day, day of week, etc. and on any periodic basis, i.e. monthly, yearly, only second Tuesdays of March.

SyzCMD/z can also make great use of the SyzMAIL/z product to send extremely complex emails that allow the use of nicknames and other special attributes.

7. Available Cellular gateways

Most carriers have “SMS gateways” which take email messages from the Internet and deliver them to their customers’ cell phones as SMS text messages. The trick is that you need to know what carrier the recipient’s phone is on — it’s not enough to know their phone number. That’s because the carrier determines what the email address of the receiving phone is going to be. For example, Cingular phones’ address are all “something@cingularme.com” while Verizon phones are “something@vtext.com.”

Sound complicated? It’s not. All you really need to do is find your carrier in the list below, and then use the pattern shown there to figure out your email address. If you want to send email to another persons’s phone, just ask them which carrier they use, and off you go!

For the major carriers, we have included a link to a page with more detailed information on how SMS works with that carrier, how much it costs, and where you can find more information. In the list below, just click any carrier’s highlighted name to find out more.

Alltel
[10-digit phone number]@message.alltel.com
Example: 1234567890@message.alltel.com

AT&T (formerly Cingular)
[10-digit phone number]@txt.att.net
Example: 1234567890@txt.att.net

For multimedia messages, use [10-digit-number]@mms.att.net
Example: 1234567890@mms.att.net

Boost Mobile
[10-digit phone number]@myboostmobile.com
Example: 1234567890@myboostmobile.com

Cricket Wireless
[10-digit phone number]@sms.mycricket.com
Example: 1234567890@sms.mycricket.com

For multimedia messages: [10-digit phone number]@mms.mycricket.com
Example: 1234567890@mms.mycricket.com

Nextel (now part of Sprint Nextel)
[10-digit telephone number]@messaging.nextel.com
Example: 1234567890@messaging.nextel.com

Sprint (now Sprint Nextel)
[10-digit phone number]@messaging.sprintpcs.com
Example: 1234567890@messaging.sprintpcs.com

T-Mobile
[10-digit phone number]@tmomail.net
Example: 1234567890@tmomail.net

Verizon
[10-digit phone number]@vtext.com
Example: 1234567890@vtext.com

Virgin Mobile USA
[10-digit phone number]@vmobl.com
Example: 1234567890@vmobl.com

Other U.S. and Canadian carriers:

Bell Canada: [10-digit-phone-number]@txt.bellmobility.ca

Centennial Wireless: [10-digit-phone-number]@cwemail.com

Cellular South: [10-digit-phone-number]@csouth1.com

Cincinnati Bell: [10-digit-phone-number]@gocbw.com

Metro PCS: [10-digit-phone-number]@mymetropcs.com or [10-digit-phone-number]@metropcs.sms.us

Qwest: [10-digit-phone-number]@qwestmp.com

Rogers: [10-digit-phone-number]@pcs.rogers.com

Suncom: [10-digit-phone-number]@tms.suncom.com

Telus: [10-digit-phone-number]@msg.telus.com

U.S. Cellular: [10-digit-phone-number]@email.uscc.net

8. Extended list of Cell Gateways

***note: replace the "number" with your phone or pin number**

Company Name	**SMTP Format**
3 River Wireless	phonenumber@sms.3rivers.net
A	
Accessyou (Hong Kong)	number@messaging.accessyou.com
ACS Wireless	phonenumber@paging.acswireless.com
Advantage Communications	10digitpagernumber@advantagepaging.com
Aio Wireless	number@mms.aiowireless.net
Airfire Mobile	number@sms.airfiremobile.com
Airtouch Pagers	10digitpagernumber@myairmail.com
Airtouch Pagers	10digitpagernumber@alphapage.airtouch.com
Airtouch Pagers	10digitpagernumber@airtouch.net
Airtouch Pagers	10digitpagernumber@airtouchpaging.com
Alaska Communications	number@msg.acsalaska.com
Aliant	number@sms.wirefree.informe.ca
Alltel	10digitphonenumber@sms.alltelwireless.com
Alltel	10digitphonenumber@alltellmessage.com
Alltel PCS	10digitphonenumber@message.alltel.com
AlphNow	pin@alphanow.net
Ameritech Paging (see also American Messaging)	10digitpagernumber@paging.acswireless.com
Ameritech Paging (see also American Messaging)	10digitpagernumber@pageapi.com
American Messaging (SBC/Ameritech)	10digitpagernumber@page.americanmessaging.net
Ameritech Clearpath	10digitpagernumber@clearpath.acswireless.com
Andhra Pradesh Airtel	phonenumber@airtelap.com
Andhra Pradesh India Cellular	number@ideacellular.net

Api4SMS	number@members.api4sms.net
Arch Pagers (PageNet)	10digitpagernumber@archwireless.net
Arch Pagers (PageNet)	10digitpagernumber@epage.arch.com
Arch Pagers (PageNet)	10digitpagernumber@archwireless.net
Assurance Wireless	number@vmobl.com
AT&T	10digit@txt.att.net
AT&T Engterprise	phonenumber@page.att.net
AT&T Free2Go	10digitphonenumber@mmode.com
AT&T Global Smart Messaging	10digitphonenumber@sms.smartmessagingsuite.com

B

B2B SMS (India)	pagernumber@b2bsms.co.in / info@b2bsms.co.in
Beepwear	pagernumber@beepwear.net
BeeLine GSM	phonenumber@sms.beemail.ru
Bell Atlantic	phonenumber@message.bam.com
Bell Canada	phonenumber@txt.bellmobility.ca
Bell Canada	phonenumber@bellmobility.ca
Bell Mobility (Canada)	phonenumber@txt.bell.ca
Bell Mobility	number@txt.bellmobility.ca
Bell South (Blackberry)	number@bellsouthtips.com
Bell South	phonenumber@sms.bellsouth.com
Bell South	phonenumber@wireless.bellsouth.com
Bell South	phonenumber@blsdcs.net
Bell South	phonenumber@bellsouth.cl
Bell South Mobility	phonenumber@blsdcs.net
Blue Sky Frog	phonenumber@blueskyfrog.com
Bluegrass Cellular	phonenumber@sms.bluecell.com
Bluesky Communications (American Samoa)	number@psms.bluesky.as
Boost	phonenumber@myboostmobile.com
Boost	phonenumber@sms.myboostmobile.com
Bouygues Telecom (France)	number@mms.bouyguestelecom.fr

Box Internet Services SMS Gateway (Switzerland)	number@sms.boxis.net
Box Internet Services SMS Gateway (Switzerland)	number@mms.boxis.net
BPL mobile	phonenumber@bplmobile.com
Bryx911	pin@alert.bryx911.com
BulkSMS.com	number@bulksms.net
Bulletin.net	internationalformatednumber@bulletinmessenger.net

C

C Beyond	number@cbeyond.sprintpcs.com
C Spire Wireless	number@cspire1.com
Cable and Wireless (Panama)	phonenumber@cwmovil.com
Carolina West Wireless	10digitnumber@cwwsms.com
Carolina Mobile Communications	10digitpagernumber@cmcpaging.com
Cellcom	number@cellcom.quiktxt.com
Cellular One East Coast	phonenumber@phone.cellone.net
Cellular One South West	phonenumber@swmsg.com
Cellular One PCS	phonenumber@paging.cellone-sf.com
Cellular One	10digitphonenumber@mobile.celloneusa.com
Cellular One	phonenumber@cellularone.txtmsg.com
Cellular One	phonenumber@cellularone.textmsg.com
Cellular One	phonenumber@cell1.textmsg.com
Cellular One	phonenumber@message.cellone-sf.com
Cellular One	phonenumber@sbcemail.com
Cellular One West	phonenumber@mycellone.com
Cellular South	phonenumber@csouth1.com
Centennial Wireless	10digitnumber@cwemail.com
Central Vermont Communications	10digitpagernumber@cvcpaging.com

CenturyTel	phonenumber@messaging.centurytel.net
Charilon Valley Wireless	number@sms.cvalley.net
Chat Mobility	number@mail.msgsender.com
Chennai RPG Cellular	phonenumber@rpgmail.net
Chennai Skycell / Airtel	phonenumber@airtelchennai.com
China Mobile	number@139.com
Cincinnati Bell Wireless	phonenumber@gocbw.com
Cingular	mobilenumber@mycingular.com
Cingular	mobilenumber@mycingular.net
Cingular	mobilenumber@mms.cingularme.com
Cingular	mobilenumber@page.cingular.com
Cingular	10digitphonenumber@cingularme.com
Cingular	10digitphonenumber@txt.att.net
Cingular Wireless	10digitphonenumber@mycingular.textmsg.com
Cingular Wireless	10digitphonenumber@mobile.mycingular.com
Cingular Wireless	10digitphonenumber@mobile.mycingular.net
Claro (Brazil)	number@clarotorpedo.com.br
Claro (Columbial)	number@iclaro.com.co
Claro (Nicaragua)	number@ideasclaro-ca.com
Claro (Puerto Rico)	number@vtexto.com
Clearnet	phonenumber@msg.clearnet.com
Cleartalk	numbers@sms.cleartalk.us
Comcast	phonenumber@comcastpcs.textmsg.com
Communication Specialists	7digitpin@pageme.comspeco.net
Communication Specialist Companies	pin@pager.comspeco.com
Comviq	number@sms.comviq.se
Connection Software (UK)	number@itsarrived.net
Consumer Cellular	number@cingularme.com
Cook Paging	10digitpagernumber@cookmail.com

Corr Wireless Communications	phonenumber@corrwireless.net
Credo Mobile	10digitcell@mypixmessages.com
Critical Alert / UCOM	number@pager.ucom.com
Critical Alert / Teletouch	10digitpagernumber@pageme.teletouch.com
CSL (Hong Kong)	number@mgw.mmsc1.hkcsl.com
CTI Mobile (Argentina)	number@sms.climovil.com.ar

D

Delhi Aritel	phonenumber@airtelmail.com
Delhi Hutch	phonenumber@delhi.hutch.co.in
Digi-Page / Page Kansas	10digitpagernumber@page.hit.net
Digicel (Dominica)	number@digitextdm.com
Dobson Cellular Systems	phonenumber@mobile.dobson.net
Dobson-Alex Wireless / Dobson-Cellular One	phonenumber@mobile.cellularone.com
DT T-Mobile	phonenumber@t-mobile-sms.de
DTC	number@sms.advantagecell.net
Dutchtone / Orange-NL	phonenumber@sms.orange.nl

E

E-Plus (Germany)	number@smsmail.eplus.de
Edge Wireless	phonenumber@sms.edgewireless.com
Element Mobile	number@sms.elementmobile.net
EMT	phonenumber@sms.emt.ee
Emtel (Mauritius)	number@emtelworld.net
Escotel	phonenumber@escotelmobile.com
Esendex (Australia)	phonenumber@echoemail.net
Esendex (Spain)	phonenumber@esendex.net
Esendex (UK)	phonenumber@echoemail.net
Esendex (US)	phonenumber@echoemail.net

F

Fido	phonenumber@fido.ca
Fido	phonenumber@sms.fido.ca
Firmensms (Austria)	0043phonenumber@subdomain.firmensms.at

G

Gabriel Wireless	10digitpagernumber@epage.gabrielwireless.com
Galaxy Corporation	10digitpagernumber.epage@sendabeep.net
GCS Paging	pagernumber@webpager.us
General Communications	number@@mobile.gci.net
German T-Mobile	number@@t-mobile-sms.de
Globalstar	phonenumber@msg.globalstarusa.com / number@g2smsc.globalstar.com
Globul (Bulgaria)	35989number@sms.globul.bg
Goa Airtel (India)	919890number@airtelmail.com
Goa BPLMobil	phonenumber@bplmobile.com
Goa Idea Cellular (India)	number@ideacellular.net
Golden State Cellular	phonenumber@gscsms.com
Golden Telecom	phonenumber@sms.goldentele.com
Google Fi	10digitcell@msg.fi.google.com
GrayLink / Porta-Phone	10digitpagernumber@epage.porta-phone.com
Greatcell	number@vtxt.com
GTE	number@airmessage.net
GTE	number@gte.pagegate.net
GTE	10digitphonenumber@messagealert.com
Gujarat Airtel (India)	919898number@airtelmail.com
Gujarat Celforce (India)	phonenumber@celforce.com
Gujarat Idea Cellular (India)	phonenumber@ideacellular.net
Guyana Telephone & Telegraph (Guyana)	number@sms.cellinkgy.com

H

Haryana Airtel (India)	919896number@airtelmail.com

Haryana Escotel (India)	9182number@escotelmobile.com
Hawaiian Telecom Wireless	10digitphonenumber@hawaii.sprintpcs.com
Helio	10digitphonenumber@messaging.sprintpcs.com
Himachai Pradesh Airtel (India)	919816phonenumber@airtelmail.com
Houston Cellular	number@text.houstoncellular.net
HSL Mobile	number@sms.haysystems.com

I

ICE (Costa Rica)	number@sms.ice.cr
Idea Cellular	phonenumber@ideacellular.net
Illinois Valley Cellular	10digitphonenumber@ivctext.com
Indigo	10digitcel@sms.indigowireless.com
Infopage Systems	pinnumber@page.infopagesystems.com
Inland Cellular Telephone	phonenumber@inlandlink.com
IPIPI	phonenumber@opensms.ipipi.com
The Indiana Paging Co	last4digits@pager.tdspager.com
Iridium	number@msg.iridium.com
I-Wireless	number.iws@iwspcs.net / number@iwirelesshometext.com

J

JSM Tele-Page	pinnumber@jsmtel.com

K

Kajeet	number@mobile.kajeet.net
Kamataka Airtel (India)	919845number@airtelkk.com
Kerala Airtel (India)	919845number@airtalkerala.com
Kerala Escotel	phonenumber@escotelmobile.com / 9847number@escotelmobile.com
Kolkata Airtel	phonenumber@airtelkol.com / 919831number@airtelkol.com
Koodo	number@msg.koodomobile.ca
Koodo Mobile (Canada)	number@msg.telus.com
Kyivstar	number@smsmail.lmt.lv

Kyivstar	number@sms.kyivstar.net / number@2sms.kyivstar.net

L

Lauttamus Communication	pagernumber@e-page.net
LMT	phonenumber@smsmail.lmt.lv
LongLines	number@text.longlines.com
Lynx Mobility (Canada)	number@sms.lynxmobility.com

M

M1 (Singapore)	number@m1.com.sg
Madhya Pradesh Airtel (India)	919893number@airtelmail.com
Maharashtra Airtel (India)	919890number@airtelmail.com
Maharashtra Idea Cellular (India)	number@ideacellular.net
Maharashtra BPL Mobile	phonenumber@bplmobile.com
Maharashtra Idea Cellular	phonenumber@ideacellular.net
Manitoba Telecom Systems	phonenumber@text.mtsmobility.com
Mas Movil (Panama)	number@cwmovil.com
MCI Phone	phonenumber@mci.com
MCI	phonenumber@pagemci.com
Mediaburst (UK)	number@sms.mediaburst.co.uk
Meteor (Ireland)	phonenumber@mymeteor.ie
Meteor (Ireland)	phonenumber@sms.mymeteor.ie
Metrocall	10digitpagernumber@page.metrocall.com
Metrocall 2-way	10digitpagernumber@my2way.com
Metro PCS	10digitphonenumber@mymetropcs.com
Metro PCS	10digitphonenumber@metropcs.sms.us
Microcell	phonenumber@fido.ca
Midwest Wireless	phonenumber@clearlydigital.com
MiWorld	phonenumber@m1.com.sg

Mobilecom PA	10digitpagernumber@page.mobilcom.net
Mobilecomm	number@mobilecomm.net
Mobileone	phonenumber@m1.com.sg
Mobilfone	phonenumber@page.mobilfone.com
Mobility Bermuda	phonenumber@ml.bm
Mobistar Belgium	phonenumber@mobistar.be
Mobitel Sri Lanka	number@sms.mobiltel.lk
Mobitel Tanzania	phonenumber@sms.co.tz
Mobtel Srbija	phonenumber@mobtel.co.yu
Morris Wireless	10digitpagernumber@beepone.net
Motient	number@isp.com
Movistar	number@correo.movistar.net
Movistar (Argentina)	number@sms.movistar.net.ar
Movistar (Columbia)	number@movistar.com.co
Movistar (Latin America)	number@movimensaje.com.ar
Movistar (Spain)	0number@movistar.net
Movistar (Uruguay)	95number@sms.movistar.com.uy
MTN (South Africa)	number@sms.co.za
MTS	10digitphonenumber@text.mtsmobility.com
Mumbai Airtel (India)	919892number@airtelmail.com
Mumbai BPL Mobile	phonenumber@bplmobile.com
Mumbai Orange	phonenumber@orangemail.co.in
My Cool SMS	number@my-cool-sms.com

N

NBTel	number@wirefree.informe.ca
NCell (Nepal)	977number@sms.ncell.com.np
Netcom	phonenumber@sms.netcom.no
Nextech	nubmer@sms.ntwls.net
Nextel	10digitphonenumber@messaging.nextel.com
Nextel	10digitphonenumber@page.nextel.com
Nextel (Argentina)	twoway.11number@nextel.net.ar
Nextel (Brazil)	10digitphonenumber@nextel.com.br

Nextel (Mexico)	phonenumber@msgnextel.com.mx
Northeast Paging	number@pager.ucom.com
NPI Wireless	phonenumber@npiwireless.com
Ntelos	number@pcs.ntelos.com

O

O2	name@o2.co.uk
O2	number@o2imail.co.uk
O2 (M-mail)	number@mmail.co.uk
OgVodafone (Iceland)	number@sms.is
Omnipoint	number@omnipoint.com
Omnipoint	10digitphonenumber@omnipointpcs.com
One Connect Austria	phonenumber@onemail.at
OnlineBeep	10digitphonenumber@onlinebeep.net
Optus Mobile	phonenumber@optusmobile.com.au
Orange	phonenumber@orange.net
Orange (Netherlands)	phonenumber@sms.orange.nl
Orange Mumbai	phonenumber@orangemail.co.in
Orange - NL / Dutchtone	phonenumber@sms.orange.nl
Oskar	phonenumber@mujoskar.cz

P

P&T Luxembourg	phonenumber@sms.luxgsm.lu
Pacific Bell	phonenumber@pacbellpcs.net
Page Plus Cellular	number@vtext / number@vzwpix.com / number@mypixmessages.com
PageMart	7digitpinnumber@pagemart.net
PageMart Advanced /2way	10digitpagernumber@airmessage.net
PageMart Canada	10digitpagernumber@pmcl.net
PageNet Canada	10digitpagernumber@e.pagenet.ca
PageOne NorthWest	10digitnumber@page1nw.com
Panacea Mobile	number@api.panaceamobile.com
PC Telecom (Canada)	number@mobiletxt.ca

PCS One	phonenumber@pcsone.net
Pelephone (Israel)	number@pelephone.net.il
Personal (Argentina)	number@alertas.personal.com.ar
Personal Communication (Russia)	sms@pcom.ru (number in subject line)
Pioneer / Enid Cellular	phonenumber@msg.pioneerenidcellular.com
Pioneer Cellular	9digitphonenumber@zsend.com
Plus (Poland)	number@text.plusgsm.pl
PlusGSM	phonenumber@text.plusgsm.pl
Pocket Wireless	number@sms.pocket.com
Pondicherry BPL Mobile	phonenumber@bplmobile.com
Powertel	phonenumber@voicestream.net
President's Choice	10digitphonenumber@txt.bell.ca
Price Communications	phonenumber@mobilecell1se.com
Primeco	10digitnumber@email.uscc.net
Primtel	phonenumber@sms.primtel.ru
ProPage	7digitpagernumber@page.propage.net
Public Service Cellular	phonenumber@sms.pscel.com
Punjab Airtel (India)	919815number@airtelmail.com

Q

Qualcomm	name@pager.qualcomm.com
Qwest	10digitphonenumber@qwestmp.com

R

RAM Page	number@ram-page.com
Red Rocket Mobile	number@txt.att.net / number@mms.att.net
Republic Wireless	10digitnumber@text.republicwireless.com
Rogers	phonenumber@pcs.rogers.com
Rogers Canada	phonenumber@sms.rogers.com
Routo Messaging	number@email2sms.routomessaging.com

S

Safaricom	phonenumber@safaricomsms.com

SaskTel (Canada)	number@sms.sasktel.com / number@pcs.sasktelmobility.com
Satelindo GSM	phonenumber@satelindogsm.com
Satellink	10digitpagernumber.pageme@satellink.net
SBC Ameritech Paging (see also American Messaging)	10digitpagernumber@paging.acswireless.com
SCS-900	phonenumber@scs-900.ru
Sendega (Norway)	number@sendega.com
Setar (Aruba)	number@mas.aw
SFR France	phonenumber@sfr.fr
Siminn (Iceland)	number@box.is
Simple Mobile	number@smtext.com
Skytel Pagers	7digitpinnumber@skytel.com
Skytel Pagers	number@email.skytel.com
Simple Freedom	phonenumber@text.simplefreedom.net
Smart Telecom	phonenumber@mysmart.mymobile.ph
SMS Broadcast (Australia)	number@send.smsbroadcast.co.au
SMS Global (Australia)	number@email.smsglobal.com
SMS Central (Australia)	number@sms.smscentral.com.au
SMSPUP (Australia)	number@smspup.com
Solavel	number@tmomail.net
Solo Mobile	10digitphonenumber@txt.bell.ca
South Central Communicaitons	number@rinasms.com
Southern LINC	10digitphonenumber@page.southernlinc.com
Southwestern Bell	number@email.swbw.com
Spikko (Israel)	number@spikkosms.com
Sprint	10digitphonenumber@sprintpaging.com
Sprint PCS	10digitphonenumber@messaging.sprintpcs.com
ST Paging	pin@page.stpaging.com
Starhub Enterprise Messaging Solution (Singapore)	number@starhub-enterprisemessaging.com

Straight Talk	number@txt.att.net / number@tmomail.net / number@mms.att.net
Strata	number@rinasms.com
SunCom	number@tms.suncom.com
SunCom	number@suncom1.com
Sunrise Communications (Switzerland)	number@gsm.sunrise.ch
Sunrise Mobile (Switzerland)	phonenumber@mysunrise.ch
Sunrise Mobile (Switzerland)	phonenumber@freesurf.ch
Surewest Communicaitons	phonenumber@mobile.surewest.com
Swisscom	phonenumber@bluewin.ch
Syringa Wireless	number@rinasms.com

T

T-Mobile	10digitphonenumber@tmomail.net
T-Mobile	10digitphonenumber@voicestream.net
T-Mobile Australia (Optus)	0number@optusmobile.com.au
T-Mobile Austria	phonenumber@sms.t-mobile.at
T-Mobile Croatia	phonenumber@sms.t-mobile.hr
T-Mobile Germany	phonenumber@t-d1-sms.de / phonenumber@t-mobile-sms.de
T-Mobile Netherlands	31number@gin.nl
T-Mobile UK	phonenumber@t-mobile.uk.net
Tamil Nadu Aircel	9842number@airsms.com
Tamil Nadu Airtel	919894number@airtelmobile.com
Tamil Nadu BPL Mobile	phonenumber@bplmobile.com
Telcel (Mexico)	number@itelcel.com
Tele2 Latvia	phonenumber@sms.tele2.lv
Tele2 Sweden	0number@sms.tele2.se
Telecom New Zealand	number@etxt.co.nz

Teleflip	number@teleflip.com
Telefonica Movistar	phonenumber@movistar.net
Telenor	phonenumber@mobilpost.no
Telestra	sms@tim.telestra.com
Teletopia SMS (Norway)	number@sms.teletopiasms.no
Teletouch	10digitpagernumber@pageme.teletouch.com
Telia Denmark	phonenumber@gsm1800.telia.dk
Telus	phonenumber@msg.telusmobility.com
Telus Mobility	phonenumber@mms.telusmobility.com
Tellus Talk	number@esms.nu
Tigo (Colubmia)	number@sms.tigo.com
TIM	10digitphonenumber@timnet.com
Ting	number@message.ting.com
Tracfone	10digitphonenumber@txt.att.net
Triton	phonenumber@tms.suncom.com
TSR Wireless	pagernumber@alphame.com
TSR Wireless	pagernumber@beep.com
Txtlocal (UK)	number@txtlocal.co.uk

U

UCOM	number@pager.ucom.com
UMC	phonenumber@sms.umc.com.ua
Unicel	phonenumber@utext.com
UniMovil Corporation (UK)	number@viawebsms.com
Union Wireless	number@union-tel.com
Uraltel	phonenumber@sms.uraltel.ru
USA Mobility	number@usamobility.net
US Cellular	10digitphonenumber@smtp.uscc.net
US Cellular	10digitphonenumber@email.uscc.net
US Cellular	10digitphonenumber@uscc.textmsg.com
US West	number@uswestdatamail.com
UTBox (Australia)	number@sms.utbox.net

Uttar Pradesh Escotel	phonenumber@escotelmobile.com

V

Verizon Pagers	10digitpagernumber@myairmail.com
Verizon PCS	10digitphonenumber@vtext.com
Verizon PCS	10digitphonenumber@myvzw.com
Verizon PCS	10digitphonenumber@vswpix.com
Vessotel	phonenumber@pager.irkutsk.ru
Viaero	number@viaerosms.com
Virgin Mobile	phonenumber@vmobl.com
Virgin Mobile	phonenumber@vxtras.com
Virgin Mobile Canada	10digitphonenumber@vmobile.ca
Vivo (Brazil)	number@torpedomail.com.br
Vodacom (South Africa)	number@voca.co.za
Vodafone Czech Republic	number@vodafonemail.cz
Vodafone Germany	0number@vodafone-sms.de
Vodafone Italy	number@sms.vodafone.it
Vodafone Japan	phonenumber@c.vodafone.ne.jp
Vodafone Japan	phonenumber@h.vodafone.ne.jp
Vodafone Japan	phonenumber@t.vodafone.ne.jp
Vodafone New Zealand	phonenumber@mtxt.co.nz
Vodafone Spain	phonenumber@vodafone.es
Vodafone UK	phonenumber@vodafone.net
VoiceStream / T-Mobile	10digitphonenumber@voicestream.net
Voyager Mobile	number@text.voyagermobile.com

W

WebLink Wireless	pagernumber@airmessage.net
WebLink Wiereless	pagernumber@pagemart.net
Webtext	phonenumber@webtext.com
West Central Wireless	phonenumber@sms.wcc.net
Western Wireless	phonenumber@cellularonewest.com
Wildmist Wireless	phonenumber@wildmist.net
Wind Mobile	10digitcel@txt.windmobile.ca

Wyndtell	number@wyndtell.com