

SUPERVISOR

Operations and Programming Reference Manual Relative to version 600.03

Contents

Introduction.....	10
What is SUPERVISOR?	10
Built-in Knowledge	13
The Perfect Operator	13
Events and efficiency	14
Benefits of SUPERVISOR	15
SUPERVISOR Components.....	15
SCHEDULE file.....	16
SUPERVISOR Libraries.....	16
MAGUS	16
OPALTAPELIB	17
OPALUSERLIB	17
METASECURITYLIB.....	17
FLEXLIB.....	17
PRINTSUPPORT	17
JOBFORMATTER.....	17
METANAPLIB.....	17
MAILLIB	18
SUPERVISOR processes	18
GRINDER	18
RECORDER	18
TAPELIBUPDATER.....	18
NAPHANDLER	18
SILOHANDLER.....	18
AFTER HALTLOAD.....	19
BACKTRACK/SCHEDULE	19
WINDOWS.....	19
Others	19
Recent Changes	20
Jun 2019 600.03 Custom Context MAXWAIT.....	20
Oct 2015 580.11 Fix bug in DBS function	20
Oct 2015 580.10 Enhanced LOG command.....	20
Sep 2015 580.07 Warn when SECADMIN priv needed.....	21
Oct 2014 570.16 Better formatting of Help Attributes	21
Jun 2014 570.07 Tapemanager Changes.....	21
Quick Command Reference Guide	22
ACCESS	22
AFTER	22
DBC	22

DEBUG	22
DELINK	22
DEFINE	22
DISP	22
DO	22
EH	22
ENTER	22
Evaluate	22
Evaluate (Query)	22
Evaluate (Stop)	22
FOR	22
FP	22
HELP	22
HO	22
HOLIDAY	23
HS	23
INPUT	23
JAMPAK	23
LC	23
LOG	23
LU	23
MAIL	23
MEMO	23
NS	23
OK	23
ONCE	23
PB	23
PDT	23
PRINT	23
QUIT	23
REC	23
RELOAD	23
REMIND	23
REMOTESPO	24
RESTRICT	24
??RUN	24
SAVE	24
SAVE(OPAL)	24
SHIFT	24
SHOW	24
Slot	24
SO	24
SS	24
SYS DIR	24
SYSLOG	24
TERM	24
TEST	24
TL	24
TP	24
UO	24
USE	24
VIA	24
VL	25
WHEN	25

WINDOW	25
WORKDAYS	25
WS	25
/	25
Getting Started.....	26
Communicating with SUPERVISOR	26
Guided Tour	27
How to Talk to SUPERVISOR.....	27
Using a Terminal	27
Using an ODT	28
Getting information	28
HELP Command.....	29
PPrint Command & Modifier	30
File Placement (FP) Command	30
Headers Open (HO) Command	33
Linked Users (LU) Command	34
Tape Directory (PDT) Command	35
Scheduling	37
Date formats	37
AFTER command	38
AFTER TRIGGER.....	38
Using the AFTER command	40
CUSTOM Command.....	42
HOLIDAY command.....	43
WORKDAYS command	43
OPAL Programming	44
PRINT ATT indents each entry by 15 characters if the attribute name length is less than 16 characters. For longer name lengths the entry is indented by one character more than the attribute length.	48
Object Classes.....	48
Expert System Program Types	49
SITUations	50
DISPlays	50
ODTSequences	50
COMmands.....	51
WHEN SLOTS	52
DEFINE Command	54
Sample OPAL Program.....	58
MEMOs	60
EVALUATE Command	61
DO Command.....	62
TEST Command	62
DISPlay (/) Command.....	63
WHEN Command	63
ENTER Command	64
Schedule or no modifier	64
Trusted	64
AFTER	65
WHEN	65
DEF	65
ODTS	65
SITU	65
DISP	65
MEMO	65
<pattern>.....	65
OPAL Example	66

Basics	70
Railroad Diagrams	70
Communicating with SUPERVISOR	71
From the ODT	71
From MARC	71
From a SUPERVISOR COMS WINDOW.....	72
From a REMOTESPO.....	75
MCSes	75
DCKEYIN Intrinsic.....	75
EXTERNALCOMMAND	76
HARDCOPY Interface.....	77
OPAL Identifiers	77
Wildcards and Patterns.....	78
<pattern>	78
Repeat Last Output (EH) Command.....	79
The FOR Modifier	79
Password handling with NODEFAULTUSE.....	80
Multiple FORs	81
FOR with AFTER	82
FOR with DEF.....	82
FOR with DO.....	82
FOR with PRINT	82
FOR with USE.....	82
FOR with WHEN	83
HELP Command	83
Next Screen (NS) Command	84
Previous Screen (-) Command	84
PPrint Modifier	85
VIA Modifier	85
Exceptions	86
Control Commands	87
Hold Schedule (HS) Command	87
QUIT Command.....	89
REC Command.....	90
<FileMap>	91
<IP and Community>	91
Names.....	91
PORT.....	93
SNMP	93
DISK	93
VAR.....	93
BYTE	93
CLOSE	94
QUIT	94
TCPIP	94
SNMP.....	94
MIB	95
DESTination.....	95
MAXPacket.....	95
MAXOctet	95
MAXMessage	95
VERsion	96
WHO	96
REMOTESPO Command	96
Set Options (SO) Command.....	98
HARDCOPY (Option Number 1).....	100
USERNOFOR (Option Number 2)	100
LOGMINIMAL (Option Number 3).....	100
NOAUTODC (Option Number 4).....	100
WAITOKTASK (Option Number 5).....	101
NODEFACTIVE (Option Number 7).....	101

STRICTODTS (Option Number 8)	101
MONITORING (Option Number 10)	102
LOGOPERATOR (Option Number 13)	102
NAMEOPERATOR (Option Number 18)	103
USDATES (Option Number 19)	103
TAPELIB (Option Number 22)	103
NOTTLOGGING(Option Number 28)	103
DEFINERESP (Option Number 30)	104
ASYNCWFL (Option Number 31)	104
NOWHENRESTART (Option Number 32)	104
NOVALIDATE (Option Number 33)	104
AUTOFS (Option Number 35)	105
NOSTATS (Option Number 36)	105
NOENTERLIST (Option Number 37)	105
AUTOTL (Option Number 38)	106
USECOPYWRITE (Option Number 45)	106
COMPRESSDEFINE (Option Number 46)	106
LOGGING (Option Number 47)	106
Send Station (SS) Command	107
TERM Command	108
TapeLibUpdater (TP) Command	111
USE Command	111
USE BDNAME	112
<BDNAME spec>	112
USE CHARGE	113
<CHARGE spec>	113
USE DESTINATION	113
<DESTINATION spec>	113
USE FAMILY	114
<FAMILY spec>	114
FOR CONFIG	114
FOR LOGS	115
FOR NAPLOGS	115
FOR SAVES	115
FOR SCHEDULE	115
FOR SYSTEM	116
FOR EXTERNAL	116
USE FILE	116
<FILE spec>	116
FOR NAPRUNLIGHT	116
FOR REBUILD	116
USE GETSCRATCH	117
USE JOB	117
<JOB spec>	117
USE LANGUAGE	118
USE MODE	118
<MODE spec>	118
USE ODT	120
USE QUEUE	121
USE SCRATCHPOOL	121
<SCRATCHPOOL spec>	121
USE SILO	121
<SILO spec>	121
USE TASK	124
<TASK spec>	124
FOR NAPSTATSLIB	124
FOR RECORDER	124
USE USER	126
<USER spec>	126
FOR EXTERNAL	127
FOR MAIL	127
FOR ODTSECURITY	127
FOR TAPELIB	128
USE NOPASSWORDS	128
USE WINDOWSECURITY	128
<WINDOWSECURITY spec>	128
WINDOW Command	129

What SUPERVISOR (WS) Command	129
System-related commands.....	131
DBControl (DBC) Command.....	131
DELINK Command	132
File Placement (FP) Command	133
FP command modifiers	134
<pattern>	134
SUMMARY	134
<chunksize>	135
MAXAVAIL	135
IO	135
Headers Open (HO) Command	136
JAMPACK Command.....	137
Linked Users (LU) Command	139
Print Backup (PB) Command.....	140
Tape Directory (PDT) Command	140
RESTRICT Command	142
Table of RESTRICT capable ODT commands.....	143
RUN Command	145
SHOW Command	146
MAP	146
GLOBALS	147
PERM.....	148
UTIL	149
LOGs.....	149
SYSDIR Command.....	150
SYSLOG Command	151
Transfer Log (TL) Command	152
Volume Library (VL) Command	152
Administration Commands.....	154
ACCESS Command	154
INPUT Command.....	155
LC Command.....	155
LOG Command.....	155
<category>	156
<DateTimeRange>	157
<Units>	157
LR Command.....	158
MAIL Command	159
RELOAD Command	160
REMIND Command	161
SAVE Command.....	162
Save Userdata	162
Save Schedule.....	162
SAVE SCHEDULE AS.....	163
SCHEDULE conversions	163
SHIFT Command	164
Defining Operator Identification	165
OPAL related commands	167
DEFINE Command	167
DEF + : Creating or Modifying Definitions	168
Version.....	169
Set/Reset.....	169
LIST	169
CODE	169
COMPRESS	169

Program Types.....	170
MEMO	170
SITUATION.....	170
ODTSEQUENCE.....	170
COMMAND.....	171
DISPLAY.....	172
OPAL Program Contexts.....	173
LOG-based contexts.....	173
Contexts in SITUations.....	174
<context>.....	175
Evaluation of SITUations	176
Non-LOG contexts.....	177
Context Notes.....	179
AFTER Context	180
CUSTOM Context.....	181
Custom Specification.....	181
Example TEST Context	183
Example FTP Context.....	183
Example XML Context.....	188
Example C74 Context.....	190
DEFINE Context	191
DBACCESS Context	192
KEYS Context.....	192
PD Context	192
MAIL Context.....	193
METALOG Context	194
RSTCONTEXT	194
SESSIONS Context.....	196
SHOWOPEN Context.....	196
SL Context	197
SMTP Context	198
RFC822 Message.....	199
MAILIOH.....	199
FLEX.....	199
File Attributes	199
CopyWrite	200
OBI	200
File Naming Conventions	200
SMTP Server MAGUS Configuration Options	200
SMTP_USERCODE	200
SMTP_AUTH.....	201
SMTP_GATEKEEPER.....	201
Security.....	201
Operations	201
SUPERVISOR DEBUG	201
Mail Clients	202
STATIONS Context.....	202
VL Context.....	202
WHEN Context	209
Execution of ODTSequences	210
OPAL Compiler Options	211
Program Sources	211
DEFining FROM files	212
Using the FOR modifier with DEF	212
DEF ? : Interrogating the OPAL Directory	213
DEF ? : Listing a Program.....	215
DEF - : Deleting a Program from the Directory	216
IN-LINE OPALs	216
<OPAL def>	216
Viewing.....	217
Saving.....	217
DISPlay Invocation Command	218
Parameters	218
MX parameters	218
PER parameters	218
MSG parameters	219
TAPEDB parameters	219
PRINTS parameters	219
VL Parameters.....	219
USER parameters	219

AFTER parameters	220
SYSTEM parameters	220
DO Command	220
DO Breakpoints	221
Terminating a DO	221
DO practicalities	222
Example of TEST & DEBUG	223
OPAL CALL DO statement	223
ENTER Command	224
EVALUATE Command	226
Evaluating a SITUation	226
<Eval limits Log contexts>	227
<range specs>	228
<datetime>	228
<duration>	228
<max evals>	229
<Eval limits After context>	229
<Eval limits others>	230
<max entries>	230
<Sort Clause>	231
Linking a SITUation to a DISPlay	231
Linking a SITUation to an ODTs	232
Stopping a SITUation	232
Interrogating Running OPAL Programs	233
WHEN states	235
MEMO Command	236
OK Command	237
PRINT Command/Modifier	237
<period spec>	238
SAVE Command	239
SAVE Programs	239
SAVE SCHEDULE	240
SAVE AFTERS	241
SAVE WHENS	241
SAVED SCHEDULE FILE SAMPLE LISTING	243
SLOT Command	244
User Options (UO) Command	245
WHEN Command	245
Monitoring with an OPAL SITUation	245
<delay spec>	245
ONCE	246
& DEBUG	246
DISP	246
PR	246
DO	246
TEST	246
DELAY	247
Activating SITUations in WHEN slots	247
How WHENS are Evaluated	248
Locking WHENS	249
Logging of WHENS and Dos	250
Resource Accounting for Compilations	251
Execution of WHENS and DOs	251
WHEN and DO Session Recovery	251
Splitting Sessions over a TL	252
Logging of Tasks	252
Logging and the TL Command	252
Caching ODTSequence code	253
Scheduling	254
Changing the System Time and Date	254

Menu Based Scheduling	255
AFTER Command	255
<time specification>	255
<time of day>	255
<trigger>	256
<day of week>	257
<holiday except>	257
<qualification by week>	260
<qualification by month>	262
<where clause>	263
A Description of the Schedule	263
Scheduling an activity	264
Deleting an activity	265
<obsolete date spec>	266
Simulating an add or delete	266
Interrogating the Schedule	267
Associating Scheduled Activities with Usercodes	269
Timestamps and Activity Postscripts	270
HOLIDAY Command	273
WORKDAYS Command	274
HOTLINE and RECORDER	277
OPAL RECORD statement	277
Recording to PORT files	278
Recording to disk files	279
Recording to disk file 0	279
Recording to disk file 11	280
Recording to disk files 50,51,52,53,54,55 (type VAR)	280
Causing RECORDER to quit	280
RECORD file names for DISK and VAR	280
RECORD file names for BYTE	281
HOTLINE program	282
Operating the HOTLINE program	282
Customising RECORDER and HOTLINE	283
COMS and HOTLINE	284
HOTLINE_MSG library entrypoint	285
Examples of using RECORD	287
SUPERVISOR Operation	288
Tailoring SUPERVISOR	291
Task Attributes	291
Renaming files	292
DEBUG Command	292
Debugging OPAL programs	293
Fault Handling	293
Security	295
Release Media	296
Advanced Opal examples	299
Example: COM HL_INFO	299
Example: COM SUP_INFO	299
Example: COM DBS	300
Example: COM Y	300
Example: COM NETWORK	301
Example: COMmand CONFIG_CHECK	302
Example: SITU and ODTs STN_NOTREADY	303
Others	304
Copyright	305
ACKNOWLEDGEMEN	305

What is SUPERVISOR?

Over the years, the power and sophistication of Unisys MCP mainframes have expanded dramatically, in terms of both hardware and software, yet the cost of the system has dropped in real terms. Meanwhile, the cost of providing operations and systems personnel to run these machines has gone up dramatically. The result has been a simplification of the workload that the machines perform under the limits of the human control feasible. SUPERVISOR was conceived as software, which would enable the system to take care of operational problems that would otherwise require human intervention. However, having opened the Pandora's box of cybernetics, SUPERVISOR was found to be a powerful tool in many other applications. It became, in effect, a 4GL for systems programming.

As far back as the late seventies, METALOGIC was quick to spot the possibilities of a new software concept: Expert Systems. This was the first practical application of Artificial Intelligence, software products that were able to "learn" the knowledge of human experts and take over the control of complex systems. At a time when such concepts rarely progressed beyond the theory stage, METALOGIC came up with a simple and elegant Automated Operations tool specifically for Burroughs (now Unisys) mainframes: SUPERVISOR.

expert system: (noun) A computer system using software which stores and applies the knowledge of experts in a particular field, so that a person using the system can draw upon that expertise to make decisions, inferences, etc.

The Oxford Dictionary of New Words, 1991 Edition

Although SUPERVISOR is not a true Expert System, the product uses a sophisticated 'knowledge' system that can provide an operations "toolbox" on the complete range of Unisys A Series computers, from LX150 to the largest CS systems.

However, despite its tremendous capabilities, SUPERVISOR is very easy to learn and use. Once the software has been installed (a simple, automated process using METALOGIC INSTALL), to the experienced Unisys operator or programmer it becomes a seamless extension to the ordinary system command mechanism. This is because, from its first inception, SUPERVISOR's own command syntax and responses have reflected those of the standard MCP.

SUPERVISOR has been developed and enhanced over the intervening years by a team of MCP experts whose own detailed knowledge of the system architecture has been built into the very foundation of the software. Yet METALOGIC has always recognised that each and every MCP site has unique operational requirements. Because of this diversity, SUPERVISOR provides powerful tools to automate operations via a combination of commands and functions programmed in OPAL, an easily-learned language for defining operational situations and remedial actions. The Metalogic support team are always happy to provide Opal examples to help customers to develop their own routines. In this way, SUPERVISOR can be tailored

to match *exactly* any installation's operational methodology by:

- learning and following the rules, techniques and expertise developed by a site, thus enforcing site policies automatically
- maintaining a schedule of jobs to be run (for years in the future if need be) and automatically initiate them
- monitoring the system, recognising exception conditions and rectifying them
- dynamically capturing information when some catastrophic system failure occurs in order to aid debugging and subsequent correction
- preventing (human) operator error
- detecting and suspending "runaway" tasks which are in infinite loops

SUPERVISOR is a rules-based Knowledge System in which the rules are DEFINED using METALOGIC's own systems language, OPAL. Each site can transform operating procedures and policies into OPAL rules that then become part of SUPERVISOR's expert knowledge base. This means that, unlike "hard coded" products, SUPERVISOR can implement any operational preferences and requirements DEFINED by the customer.

As part of the standard SUPERVISOR software installation, METALOGIC provides a complete set of skeleton OPAL programs that enables users to quickly make the most of their systems. These examples can be loaded from software release tapes just as they are, or they can be amended to suit particular site needs. Also included with the released software are some tutorial examples to be used in conjunction with [Getting Started](#) later in this manual.

To give some idea of the capabilities of SUPERVISOR, here are some common solutions successfully implemented by existing customers:

- When lack of main memory led to sudden performance degradation, SUPERVISOR could prevent thrashing by precise real-time parameter control and reduction of unused memory.
- Poor terminal response times were improved by getting SUPERVISOR to allocate the CPU preferentially to short or I/O bound tasks.
- At one site where overnight jobs were incomplete due to "SECTORS REQUIRED" hangs, SUPERVISOR monitored trends in space usage to predict future demand. Since METALOGIC JAMPACK was available, the information was used to solve the space hangs and reduce fragmentation without operator intervention.

JAMPACK is a faster and more flexible alternative to the Unisys SQUASH facility. JAMPACK runs up to 20 times faster than SQUASH, frees sectors more reliably, and can compress until a specific number of sectors has been freed.

- A new level of active system security instituted to detect, record and prevent dangerous or unauthorized use of the machine. Each time a task is started or a user signs on, SUPERVISOR ensures that if privileged it is known and conforms to site policies.. If not, any task or session is terminated and the violation is logged.
- Since both tapes and tape units often need considerable operator attention, SUPERVISOR has many built-in capabilities to control and monitor the tape subsystem and the tape library. One customer prevented operators purging tapes with valid data by implementing the METALOGIC Tape Library Database Control System. Then mechanisms were set up using SUPERVISOR which physically prevented operators purging in-use tapes.
- One site had users who would start resource-intensive tasks (such as LINC) then leave their terminals with the tasks using up memory. SUPERVISOR was used to detect these "machine hogs" and notify the operators to take action.

METALOGIC uses this kind of customer feedback to continually update and enhance the standard SUPERVISOR software. Solutions that have a general application are widely disseminated between customers in the form of OPAL programs and METALOGIC provides regular upgrades to take advantage of the latest levels of MCP released by Unisys.

Three powerful sub-systems combine to provide SUPERVISOR's extensive capabilities. First, there are the built-in commands and functions for eliminating many frequently-seen operational bottlenecks. These include :

- Automated Halt Load, and Restart handling, including restoration, logging, analysis, and reports.
- An operational database to provide a HELP facility and eliminate most of the paper that often chokes the system console area.
- Improvements on some of the standard system commands for better efficiency and more operational convenience.
- Automatic backup of critical system files and, where appropriate, restoration on demand.

Secondly, SUPERVISOR incorporates a time scheduler which allows reliable and flexible scheduling – hours, days or even years in advance. In a standard operating environment, commands to the system via the ODT are always executed immediately. The Scheduler in SUPERVISOR introduces a new possibility – the delaying or scheduling of system commands for a specific time or range of times in the future.

Last, but by no means least, is METALOGIC's **OP**erations **A**lgorithmic **L**anguage – OPAL. Since OPAL is similar to Unisys Work Flow Language (WFL) as well as the standard system (ODT) commands, it is easy to learn. New system commands can be programmed in minutes and these can either be executed manually or used directly by OPAL programs. This allows operators and operations management to

formulate site policies in a common, unambiguous language and have those rules applied automatically.

The main features of OPAL are:

- access to the built-in knowledge base of MCP information through keywords called ATtributes
- a WFL-like syntax with full arithmetic and Boolean expressions as well as an enhanced string handling capability
- a type of program called a SITUation, which is used to detect events or recognise conditions
- another type of program called an ODTSequence, which generates a sequence of system commands and executes them
- a report writing program type called a DISPlay, which can format information and send it to the ODT or a remote terminal

In addition to the OPAL program types, the user can also build on-line HELP using MEMOs, which can give detailed information about site-specific SITUations, ODTSequences or DISPlays.

Built-in Knowledge

In order to be a true operations expert, SUPERVISOR needs information about what is happening on the system to make correct and informed decisions. These elements of system information are obtained from the MCP and are automatically available to an OPAL Program. OPAL views the system as containing Objects classes of various types. Each different Object class or Context is DEFINEd by a set of ATtributes. Here are some examples of Object classes:

SYSTEM: information about the system as a whole, such as the amount of memory in use, segments available on a particular pack, percentage of false idle, and so on.

MX: details of tasks currently in the mix, such as core in use, processor time used, whether a program is privileged, the text of wait messages, etc.

PER: peripheral information, such as number of I/Os and errors on a given device since the last Halt Load, the serial number of a tape, etc.

There are many others: MSG, LOG, LOGON, LOGOFF, FILEOPEN, FILECLOSE, NAPLOG, TAPELABEL, PRINTS, SECURITY, LOGEOJ, etc.

The Perfect Operator

In designing SUPERVISOR, the METALOGIC team took a fundamental decision to come up with a system that was as "natural" and instinctive as possible. Any intelligent organism has sensory inputs that enable it to detect changes in its environment. Humans can see, smell, taste, hear and touch things around them. Based on these inputs, we can either take action or we can communicate what we

have deduced from our senses. For example, while driving a car, many rapid sensory inputs govern our reactions. We see a red light and stop; a dog runs into the path of the car and we brake; someone cuts in dangerously and we swerve, at the same time sounding the horn, flashing the lights, and giving vent to our anger in a selection of well-worn Anglo-Saxon phrases.

This natural division of behaviour – information input, direct action, and communication – is mirrored in SUPERVISOR. It looks for a given set of circumstances (known as a SITUation) either by requesting event notices from the MCP or by sampling the state of the system. The particular circumstance to be found can be a job entering the mix, a task completing abnormally, a waiting entry that needs pack space – any of the kinds of things a human operator would be on the look-out for.

Once the SITUation has been detected, SUPERVISOR can respond. This can take the form of a communication by sending a message to the operator (a DISPlay), or can be direct action (an ODTSequence). This behaviour is readily understandable and writing the appropriate SITUations, ODTSequences, and DISPlays is mere common sense, dictated by our own day-to-day experience.

Events and efficiency

For most of its functions, SUPERVISOR captures copies of event notices which are normally internal to the MCP. This provides an enormous advantage over other supervisory or performance measurement programs which work by sampling the system on a periodic basis. SUPERVISOR is more efficient because, in many cases, there is no overhead in waiting for event notifications and, once notified, it can act immediately rather than awaiting the next sample. Indeed, in some circumstances this fast response is not just convenient but critical. For example, SYSTEM/MAKEUSER should only be run by Security Administration personnel. In detecting unauthorised runs, SUPERVISOR must act before MAKEUSER has the opportunity to affect the USERDATA File. Only an event-driven mechanism can accomplish this.

Given the power and flexibility of SUPERVISOR and OPAL, the user must have an awareness of how information is retrieved by the software and this is discussed later in the manual. Care must be taken in the construction and implementation of user-written SITUations, ODTSequences and DISPlays so that event-type contexts are used in preference to system sampling (although sometimes this is the only method available for gathering the required information).

Because SUPERVISOR has been developed by MCP specialists, the software uses the very minimum of systems resources. Many tasks are performed in "sessions" similar to CANDE; others are fired up as and when needed. Some of the commands available in SUPERVISOR (such as PDT) are improved, much faster versions of similar standard system commands and functionality in the software is modularised as far as possible. For example, much of the common functionality resides in a system library, MAGUS, which is shared by other METALOGIC products.

Benefits of SUPERVISOR

Although the facilities provided by SUPERVISOR are primarily directed towards the benefit of the Operations Department of an installation, its impact is more general.

From the point of view of an EDP Manager, the main advantage is that proper use of SUPERVISOR eases involvement in the day-to-day work of the site. This occurs because SUPERVISOR's reporting features supply otherwise unobtainable information, thus allowing more informed decision and policy making. Since the process of definition of SITUations in effect implements a site policy, increased delegation of routine decision making becomes possible.

For the Operations Manager, the advantages of SUPERVISOR are that there is increased control on the work floor with less personal involvement. Also, SUPERVISOR's management reports greatly reduce the time needed to monitor the performance of personnel and hardware. Operators are freed from much routine decision making and can concentrate their attention on rare exception events.

The process of describing SITUations and ODTSequences to SUPERVISOR will often have beneficial side effects. In particular, all personnel involved will have a common language. Training of operators is greatly eased if site policies are described in terms of how SITUations are handled. Since the Scheduler includes, as a subset, the possibility of automating all regular job scheduling, one traditionally time-consuming aspect of Operations is greatly simplified.

For the Systems Programmer, SUPERVISOR is a tool of great power. Often a major part of a Systems Programmer's time is devoted to determining how an exception event occurred. SUPERVISOR's logging facilities, and the OPAL language, allow not only detection of exception conditions as they occur, but also the ability to take preventive action. Judicious use of these features allows the Systems Programmer to provide more informed advice to management, with the documentary proof so often lacking where operational procedures are involved.

In short, SUPERVISOR allows technicians and management to increase their control over a site's operation at all levels.

SUPERVISOR Components

SUPERVISOR is a continuously running DCALGOL program with a fairly small static core size and a small dependent NEWP library, which is SL-ed with the FUNCTIONNAME of MAGUS. Other subordinate tasks are fired off for specific functions as needed. The base stack is an MCS (**M**essage **C**ontrol **S**ystem), a type of program, which is granted special privileges by the MCP. Apart from the time it is executing initialisation procedures, it is largely event driven therefore the facilities it provides cost resources only in proportion to their use.

SUPERVISOR will add itself as an Automatic Initiator (by doing an AI system command), so that the MCP will always initiate it following a Halt Load.

SUPERVISOR "arranges" the AI list such that Supervisor will be the first job to execute.

All MCS programs are granted privileged access to certain MCP intrinsic functions. SUPERVISOR makes extensive use of GETSTATUS, SETSTATUS and SYSTEMSTATUS to monitor and control its environment. The intercom queue facility (of the MCP) and DCKEYIN intrinsic are used to communicate with the system CONTROLLER. Intercom queues are used by SUPERVISOR to communicate with other MCSes and HARDCOPY.

SCHEDULE file

SUPERVISOR uses a disk file, called "*SCHEDULE", to store all information necessary to support its scheduling capabilities (maintained by the AFTER and WHEN commands), the status of options, and other control information. The FILEKIND of the SCHEDULE is set to RECOVERYFILE preventing unauthorised modifications.

Following any sort of restart, this file must be found by SUPERVISOR without modifying the codefile (due to the software's method of initiation) and is therefore required to be on the same family as JOBDESC (set by the DL JOBS system command) or that specified by the TT USE FAMILY.. FOR SCHEDULE command. If not available on either of these families, as in the case of a Cold Start, SUPERVISOR will look on the DL JOBS family and, if it is still not found, will create a new file.

Because the SCHEDULE file has to maintain information possibly years in advance, yet has to be continuously on-line, address check and checksum are performed on all its I/O operations. Security backup of the file is integrated into SUPERVISOR and both binary and de-compiled symbolic copies may be saved.

From time to time, Metalogic may alter the structure of the SCHEDULE file; in such cases an internal 'schedule version' number is incremented and once converted, it is not possible to revert to an earlier version of SUPERVISOR. For example, SUPERVISOR displays these messages during a conversion and restart:

```
SUPERVISOR:CONVERTING SCHEDULE TO VERSION 16
SUPERVISOR:SCHEDULE CONVERSION DONE
SUPERVISOR:SCHEDULE IS 95% AVAILABLE (2010 SEGMENTS IN USE)
```

SUPERVISOR performs extensive checks on the SCHEDULE file during initialisation and if any integrity problems are found, the current file is changed to *OLDSCHEDULE/<timestamp> and attempts to load a backup copy of the SCHEDULE is performed (see SAVE SCHEDULE command). If this fails, SUPERVISOR will create a new SCHEDULE file.

SUPERVISOR Libraries

MAGUS

MAGUS is a support library for all METALOGIC products, including: SUPERVISOR, FLEX, JAMPACK, DBCONTROL and COPYWRITE. SUPERVISOR uses MAGUS

for determining its licence expiry date (if any) on the current machine, and also for implementing several specialist commands, including HO and LU. SUPERVISOR calls MAGUS with the library attribute LIBACCESS set to BYFUNCTION. This library must therefore be SL-ed to the code file chosen by each installation. However, its codefile title must have the text "METALOGIC" within it.

OPALTAPELIB

All access to the tape library database, METATAPELIB4, passes through an SL-ed Library with the name of OPALTAPELIB. Like MAGUS, its codefile title must have the text "METALOGIC" within it.

OPALUSERLIB

OPALUSERLIB implements an OPAL function, called USERFN, which allows the user to add his/her own tailored functions to OPAL scripts. Please see the **Metalogic OPAL Reference manual** for more details.

METASECURITYLIB

This SL-ed library, if present, validates user input from SUPERVISOR COMS windows/Remotespos and also checks operator sign-on. See [Security](#) for more details.

FLEXLIB

A small collection of OPAL attributes, which start with "VS" in the SYSTEM VL subset, allow the retrieval of information from the FLEX Volume Statistics file using interfaces in the Metalogic FLEXLIB SL-ed library. This file is maintained by FLEX.

PRINTSUPPORT

SUPERVISOR will link to the Unisys PRINTSUPPORT SL-ed library to support the PRINTS OPAL object.

JOBFORMATTER

This Unisys SL-ed library is used to format the output of the LOGTEXT attribute, which is valid for all objects in the LOG super object class.

METANAPLIB

This Metalogic library supports OPAL attributes of the SYSTEM NAP family and the NAPLOG event context. The implementation assists with the return of statistical information about how a Unisys NAP (Network Application Platform) system is running and the reporting of network events in real-time.

MAILLIB

This Metalogic library supports the OPAL MAIL statement that allows E-mail messages to be generated by an Opal program and sent via a SMTP gateway.

SUPERVISOR processes

During normal running, SUPERVISOR has many and various sub-tasks active. Some are short lived while others can remain in the mix for long periods. The major ones are:

GRINDER

The GRINDER task handles requests from SUPERVISOR that may take a considerable amount of time to complete, such as the compilation and starting of WFL jobs, and the OPAL KEYIN, USERFN and COMS functions.

RECORDER

RECORDER controls the programmatic update of report information to disk files and passing of event information to HOTLINE clients, both using the OPAL RECORD statement. Please see **Chapter 9: HOTLINE and RECORDER** for more information.

TAPELIBUPDATER

The TAPELIBUPDATER process provides the conduit for operator access, using the TP command subset, into the Metalogic Tape Library Database (TRIM) system. This task also calls OPALTAPELIB entrypoints to automatically update the database with new tape creation notices, purges and general tape usage.

NAPHANDLER

The NAPHANDLER task is responsible for handling all access to the NAP Statistics and NAPSUPPORT libraries to support OPAL NAP attributes and NAPLOG context.

SILOHANDLER

SILOHANDLER provides support for the TK command subset (supports the Unisys/StorageTek Tape robot systems) and the TM command subset (for integration with DSI's TAPEMANAGER product). This process also handles OPAL attributes such as CARTINSILO.

AFTER HALTLOAD

SUPERVISOR invokes this process during program initialisation to action specific activities scheduled to only take place after a Halt-Load (set by the AFTER + HL command).

BACKTRACK/SCHEDULE

The BACKTRACK/SCHEDULE process is invoked each time SUPERVISOR detects that scheduled AFTER activities have been missed, after a restart or outage. It's action depends on the setting of TT HS and TT USE MODE.. FOR BACKTRACK.

WINDOWS

All terminal activity via Supervisor's COMS windows is handled by the SUPERVISOR/WINDOWS task. Previously, the GRINDER process handled all Supervisor-COMS activity.

Others

SUPERVISOR processes many housekeeping tasks to perform the following:

- save/decompile/print the SCHEDULE file
- save one or more OPAL programs from the SCHEDULE file
- save and process SYSTEM/SUMLOG
- compile one or more OPAL program(s)
- offload TAPEDB and Log OPAL processing to EVALREADER tasks
- compile and process scheduled WFL jobs

Other tasks include external codefiles such as *METALOGIC/JAMPAK (reduces disk fragmentation and handles SECTORS REQUIRED waiting entries), SYSTEM/FILEDATA, SYSTEM/LOGANALYZER.

Recent Changes

This section of the manual highlights important new features available in the OPAL compiler and run-time 'machine', in reverse-chronological order. The information will be regularly updated every time the content changes.

Jun 2019 600.03 Custom Context MAXWAIT

When a Custom Context is linked, a dependent asynchronous task is initiated to perform the LINKLIBRARY, to avoid hanging SUPERVISOR. If the LINKLIBRARY is not completed within 1 second, the task is DSED. To allow that wait timeout to be modified, the syntax for invoking a Custom Context has been enhanced. The syntax currently allows a "... ,<eval limit>]" after the library specification. The new syntax is "... ,<modifier list>]", which includes that syntax.

```
<modifier list> ::= <modifier> | <modifier list> , <modifier>
<modifier> ::= <eval limit> | <wait limit>
<eval limit> ::= <integer> | MAXEVAL = <integer>
<wait limit> ::= MAXWAIT = <integer>
```

Oct 2015 580.11 Fix bug in DBS function

If the user reported in the TT USE command for 'USER for TAPELIB' did not exist then the DBS function would fail, generating a Security violation. This has been corrected.

The Copy generated by the TT SAVE USERDATA or TT RELOAD USERDATA would fail if the Supervisor usercode did not have Secadmin privilege and the system SECADMIN option was 'authorized'.

Supervisor now checks this and does not allow the command if the secadmin privilege is not set.

The help for TT SAVE and TT RELOAD has been brought up to date.

Oct 2015 580.10 Enhanced LOG command

If the response to a LOG command is truncated, due to the record limit, the record limit is now reported along with the date and time of the last record scanned.

A new 'time spec' option MORE has been added which allows a new log command to continue from where the previous one stopped.

Ex.

```
LOG MSG + will stop after 500 entries have been found.
LOG MSG MORE will continue from the minute where the previous
command stopped.
```

Sep 2015 580.07 Warn when SECADMIN priv needed

If the TT TL command is used and the log files should be moved (DL LOGS Neq Supervisor's Family for Logs) the Supervisor usercode must have SECADMIN privilege to be able to copy the Security log. If this is not the case then the copy will abort with a security violation.

Although it would be possible for Supervisor to give the Usercode Secadmin privilege, it was decided that this was best left in control of a human secadmin.

Supervisor will now return an error on a TT TL command and TT USE Family command if it would mean moving log files when the Supervisor usercode does not have Secadmin privilege.

Ex

```
Usercode SUPERVISOR must have SECADMIN privilege when Logs are to be moved
```

Oct 2014 570.16 Better formatting of Help Attributes

The Help Attributes command when passed a pattern, lists the attributes in two columns. Depending on the length of the longest attribute name to be displayed and the width of the screen, some data could be truncated. Now if either an entry in the first column or the second column would be truncated, only the entry in the first column is show.

Previously PRINT ATTRIBUTES would indent each entry by the length of the longest attribute name. Now only entries with a name longer than 15 characters are indented more than 15 characters and then only by one character more than the attribute length.

Jun 2014 570.07 Tapemanager Changes

Related Tapemanager changes. Added new debug code controlled by TL_DEBUG config variable.

Quick Command Reference Guide

Supervisor commands are grouped into different sections of the manual. This section gives a brief summary of each command. Clicking on the command name will take you to its description in the manual.

[ACCESS](#)

Change operator accesscode

[AFTER](#)

Add, delete or interrogate a scheduled activity

DBC

Send command to METALOGIC/DBCONTROL

DEBUG

Control debugging facilities with SUPERVISOR

DELINK

Delink from attached Libraries

DEFINE

Maintain SUPERVISOR's library of OPAL programs

[DISP](#)

Invoke a DISPlay and send the results to the terminal

DO

Run or stop ODTSequence without a SITUation

EH

Re-display the last screen output by SUPERVISOR

ENTER

Update the *SCHEDULE file from a disk file

EValueate

Evaluate, interrogate or terminate a SITUation

[Evaluate \(Query\)](#)

Interrogate running Opal

[Evaluate \(Stop\)](#)

Terminate Running Opal

[FOR](#)

Provide security to scheduled activity, WHEN, DO, etc.

FP

File Placement

HELP

Provide information about TT Commands and attributes

HO

Report on disk file headers open on a family

HOLIDAY

Show, add or delete dates from current holiday list

HS

Interrogate, set, reset SUPERVISOR's Hold Schedule Flag

INPUT

Pass the contents of a disk file to CONTROLLER where they are interpreted as ODT Commands

JAMPAK

Run program METALOGIC/JAMPAK

LC

Close or enter a comment into the SupervisorLog

LOG

Examine the Supervisor Log.

LU

Show users linked to LIB or DBS stack

MAIL

Interface to the Metalogic Mail Library

MEMO

Display the contents of an OPAL MEMO

NS

Next Screen

OK

Used to enter an acknowledgement

ONCE

Monitor for a condition with a SITUation, then execute an ODTSequence or DISPlay once only

PB

Run SYSTEM/BACKUP

PDT

Replacement for system command "TDIR SPO"

PRINT

Get information in hard copy form on a line printer

QUIT

Force an orderly immediate termination of SUPERVISOR

REC

Interface to Supervisor's Recorder program

RELOAD

Reload a backed up SCHEDULE or USERDATA file

REMIND

Used to supply prompt message to the operator

REMOTESPO

Analogous to the system REMOTESPO command

RESTRICT

Restricts ODT commands

??RUN

Like the ODT Primitive Run Command.

SAVE

Save a copy of *SCHEDULE or Userdata

SAVE(OPAL)

Extract from schedule to a symbolic file

SHIFT

Notification to SUPERVISOR of a shift change

SHOW

Info on internal performance and SCHEDULE file

Slot

Debug info for an active WHEN slot

SO

Interrogate, set or reset SUPERVISOR's options

SS

Send Message to a Supervisor window

SYSDIR

Run SYSTEM/FILEDATA (like ODT DIR)

SYSLOG

Run SYSTEM/LOGANALYZER

TERM

Like the ODT TERM command, so not accepted at an ODT

TEST

Used to debug an ODTSequence

TL

Releases the System SUMLOG and initiates the USE FOR TL job for each SUMLOG/= file found

TP

TRIM database interface. Also see TRIM Manual.

UO

Interrogate, set or reset SUPERVISOR's User Options

USE

Interrogate, set or reset SUPERVISOR's run time parameters

VIA

Make command appear as if it came from a different ODT

VL

Volume add or delete on a single reel magnetic tape

WHEN

Monitor for a condition with a SITUation, and execute an ODTSequence or DISPlay each time it occurs

WINDOW

Info on which windows are open to SUPERVISOR

WORKDAYS

Show, add or delete certain Saturdays or Sundays as working days

WS

Gives the version of SUPERVISOR and installed options

/

Invoke a DISPlay and send the results to the terminal

Getting Started

This chapter allows the new SUPERVISOR user to take a quick "guided tour" around the features available in the software. However, this is only meant to give a flavour of SUPERVISOR's abilities and should not be treated as a comprehensive reference. In [Advanced OPAL Examples](#), a selection of OPAL programs, varying in complexity, have been provided to show some of the techniques you can use when writing OPAL programs. It is worthwhile trying them out now even if you don't completely understand how they work; you can always come back to them at a later stage.

It is assumed that you have already successfully installed SUPERVISOR. If not, please refer to the procedures detailed in Metalogic's **Software Installation Guide**.

SUPERVISOR will issue a waiting entry after initialisation, or the load of a new day's Schedule at midnight, if the Metalogic licence keys for any of its modules have expired or are due to expire within the next 30 days. Keys for Supervisor, Trim and Advanced Scheduling are all checked.

The following waiting entry shows that the Advanced Scheduling module will expire in 6 days:

```
---Job--Task-Pri---Elapsed----- 1 WAITING ENTRY -----
*60353/60362 95 :04 (SUPERVISOR) (SUPERVISOR)SUPERVISOR/LICENCEKEY/WARNING
    SUPERVISOR:ACCEPT:Metalogic Keys:ADVSCHED:6 days left.
```

Note that even if a module appears as EXPIRED, it will continue to operate normally until the MAGUS library is restarted.

Full details of commands and their syntax are to be found in later in this manual. The OPAL language is fully documented in the companion volume: METALOGIC OPAL Programming Reference Manual. Users are assumed to be familiar with Unisys system (ODT) commands and Work Flow Language (WFL). For details of these, refer to the following Unisys A Series publications:

- **System Commands Operations Reference Manual**
- **Work Flow Administration and Programming Guide**
- **Work Flow Language (WFL) Programming Reference Manual**

Before you begin, you will need access to one of the input sources that SUPERVISOR recognises. You should also make sure that the SUPERVISOR software has been properly installed and is running on your system.

Communicating with SUPERVISOR

SUPERVISOR is designed to integrate with the standard MCP/ODT environment so that it appears as part of the operating system software rather than an application program. In order to try out the examples shown in this chapter, you need access to one of the input sources from which SUPERVISOR will accept commands. The most common examples are:

- system consoles (ODTs).
- datacom station or ODT under a COMS window, which is an MCS window under SUPERVISOR.
- a datacom station or ODT running MARC to enable use of the "TT" Directive.
- a datacom station running SUPERVISOR's REMOTESPO, which is opened as a DIRECT FILE with KIND=REMOTE.

The recommended access method is via a Supervisor window under COMS

In each case, any response generated by SUPERVISOR will be returned to the originating source. SUPERVISOR responses resemble those of the system CONTROLLER. In the case of system consoles (ODTSequence), the terminal attributes (set with the TERM system command) are used by SUPERVISOR to generate responses. Responses to datacom stations are paged using the attributes specified by the MCS forwarding the message. Responses to SUPERVISOR's own REMOTESPO stations are paged using the NDL terminal attributes.

Guided Tour

This section gives a sample of the capabilities and features of SUPERVISOR. After following this tour, you should be in a position to begin exploring the capabilities within SUPERVISOR on your own. You will find more detailed explanations and more examples for all the commands in subsequent sections.

It is assumed SUPERVISOR has already been installed on your system using the META/INSTALL program. All the OPAL example programs used but not listed in this chapter will be available in a file on the release tape called:

OPALS/SUPERVISOR/GETTINGSTARTED

How to Talk to SUPERVISOR

There are various ways to interact with SUPERVISOR; we will discuss two of the most common access methods:

Using a Terminal

The normal, and simplest way to communicate with SUPERVISOR is using the SUPERVISOR WINDOW in COMS. The normal name for this window is SUPERVISOR, so enter:

?ON SUPERVISOR

If this does not succeed in putting you on the SUPERVISOR WINDOW, see your System Administrator. If you ARE the System Administrator, see [Coms Window](#) for details on setting up the SUPERVISOR WINDOW. This window simulates an ODT, but in order to distinguish ODT input from SUPERVISOR input, some ODT

commands may require a prefix of "ODT".

Using an ODT

To communicate with SUPERVISOR from the machine room we would normally use an ODT. All SUPERVISOR commands are prefixed by "TT". The TT command on an ODT (in a non-SUPERVISOR system) is a simple "Text Test" function whereby any text after "TT" is written to the SUMLOG.

SUPERVISOR "intercepts" these TT inputs by requesting notification from the MCP of certain log entries. When a log entry of type "TT" is found, SUPERVISOR retrieves the text and treats it as an input command. Meanwhile, the following message is returned to the originating ODT:

```
TEST FUNCTION
```

Getting information

There are a number of SUPERVISOR commands available for getting information about SUPERVISOR and the system. These are similar to standard system commands where a single input produces one or more responses. For instance, to find out the level of SUPERVISOR software, use the WS command:

```

WS
----- METALOGIC SUPERVISOR Mix 63010 -----
SUPERVISOR Version 53.999.35 Compiled at 16:22:14 on 18/03/08
Attributes Version 53.530.13 Modified at 16:32:32 on 14/03/08
OPAL Version 53.530.11 Compiled at 16:33:54 on 14/03/08
MAGUS Version 53.530.05
MCPs from Version 48.0 to 53.999 are compatible with this codefile

---- LICENSE DETAILS ----
Installed modules : SUPERVISOR Expiry date : 02/06/2008
                   LOG CONTEXTS Expiry date : 02/06/2008
                   TRIM Expiry date : 02/06/2008
                   ODT CONTROL Expiry date : 02/06/2008

---- RUN-TIME DETAILS ----
Maximum WHEN slots : 40
Task PRIORITY : 50
Log session : 63016
Compile-time option DEBUG is SET
MCP commands which match Supervisor commands must be prefixed by ODT

Window S/1 at POWEREDGE

```

Similarly, you can check the current SUPERVISOR environment by interrogating the USE settings:

```

USE
----- USER and FAMILY settings -----
USER : SUPERVISOR USER for TAPELIB : TAPELIB
USER for ODTSECURITY : META USER for EXTERNAL : META
USER for MAIL : META FAMILY for NAPLOGS : VMSA
FAMILY for EXTERNAL : DEV FAMILY for SAVES : DEV
FAMILY for LOGS : DEV FAMILY for SYSTEM : DISK
FAMILY for SCHEDULE : DISK
----- TASK and FILE settings -----
FILE for REBUILD : (META)SYMBOL/SCHEDULE ON DEV
JOB after TL : (IPP)TL/HANDLER ON DEV
TASK for RECORDER : Not specified
TASK for NAPSTATSLIB : *NAPSYSTEM/NSL ON VMSA
FILE for NAPRUNLIGHT : *NAP/RUNNING/LIGHT ON VMSA
----- Miscellaneous settings -----
QUEUE : 255 BACKTRACK mode : AUTO
ODT : 1 LANGUAGE : ENGLISH
WINDOWSECURITY : MAXIMUM PASSWORD VALIDATION : NOT ACTIVE
DESTINATION : MAIL:bobandian
SILO SUPPORT : UNISYS

Window S/1 at POWEREDGE

```

HELP Command

SUPERVISOR has on-line HELP available to the user. The HELP command provides information about TT Commands and ATtributes. Enter the following to display the home page of available topics:

TT HELP

SUPERVISOR's response has more than one screen of information, so a Next Screen command TT NS (or TT + on a Window Station) is inserted on line 1, position 1 with the cursor parked after the last character. Press <Transmit> to get the next page of a multi-page response.

To display the detailed Help Information for a TT command, enter

```
TT HELP <command>
```

For example:

```
HELP EH
----- SUPERVISOR HELP -----
EH Command:
  The EH Command, requests SUPERVISOR to repeat the last output
  it made to the requesting terminal.

                        I--- EH ---I

Window S/1 at POWEREDGE
```

The on-line help gives a simple explanation of the command and a railroad diagram of the command's syntax.

PRint Command & Modifier

Sometimes it is convenient to produce a hardcopy of the displayed output from SUPERVISOR. Any SUPERVISOR command may be prefixed with PRINT or PR to direct the output to the printer. The syntax for redirecting output from the screen to the printer is

```
TT PR <SUPERVISOR Command>
```

This can be directed to a local printer of your choice by setting the printout DESTINATION via the USE command:

```
TT USE DESTINATION <destination>
```

By default, printouts are produced under the SUPERVISOR usercode (default = SUPERVISOR) and with a "dummy" chargecode of PRCMD (for PRint command). For example, to print the output from the above HELP command enter:

```
TT PR HELP EH
```

To print the entire HELP file simply enter

```
TT PRINT HELP =
```

SUPERVISOR will respond with "REPORT PRINTED BY JOB <jobnumber>" when the report has been generated.

File Placement (FP) Command

SUPERVISOR has several "built in" commands that provide more information than standard MCP commands and also run more efficiently. For instance, it is advisable to periodically analyse disk usage. To obtain information regarding how disk space is being utilised you can use the standard DU system command.

The Unisys DU command produces an output similar to:

```

DU ON DISK
Disk Usage Report for DISK (Entire Family)

Available:  -- Sectors --  ----- Bytes -----  Percent
In Use :    14,681,021      2,642,583,780      50 %
Capacity :   14,446,168      2,600,310,240      50 %
          :    29,127,189      5,242,894,020     100 %

Available Space Analysis

Areasize  -- Sectors --  ----- Bytes -----  Percent
Largest :    6,236,459      1,122,562,620      21 %
< 504:      431,027         77,584,860       1 % (3,182 Areas)
> 503:     14,249,994      2,564,998,920     49 % (391 Areas)

28,056 areas of 504 sectors could be allocated

Window S/1 at POWEREDGE

```

Alternatively, you can use the SUPERVISOR FP command.

FP (File Placement) displays disk statistics, and reports the number of available sectors for each pack on the system, the number of reads and writes, and the amount of data transferred to and from the unit. By utilising this command, the operator can determine how much space is available on each disk and how heavily it is used.

FP has several optional modifiers allowing the user to select individual families using wild cards, selecting thresholds or changing the format of the reports.

To view the default report enter:

TT FP

SUPERVISOR's response will look something like this:

```

fp
Unit  Familyname      #x  DiskSize  TotAvail  %    >504  % Open  Err
-----
507  CDIMAGE           #1   34952554  25675366  73    25675366  73    0    0
501  DEV               #1   17476277  15106011  86    15105153  86    29   0
506  DEVAUDIT          #1     5915    1383     23    1383     23    0    0
504  DEVDLPACK         #1     5915    1887     32    1887     32    0    0
500  DISK              #1   17476277  6134824- 35    6019183  34   236   0
508  DLPACK            #1     5915    1887     32    1887     32    0    0
503  DRPACK            #1     5915    1887     32    1887     32    0    0
505  LINC17            #1     5915     207      3      0        0    0    0
509  TEST5             #1     5915    1887     32    1887     32    0    0
510  TEST5             #2     5915    5887+100  5887    100    0    0
511  TEST5             #3     5915    5887    100    5887    100    0    0
      TEST5           #3     17745    13661+  77    5887     33    0    0
-----
** TOTALS **           69952428  46937113  67    25675366  37   265   0

Window S/1 at COURSE2MCP

```

Note the plus sign (+) next to the Total Available sectors column; this indicates that the space available is increasing. If a minus sign (–) appears, sectors are decreasing. If blank appears, this indicates that it is either the first time this information was displayed or that the information has not changed since the last

display. The column labelled >504 indicates the number of sectors available in areas equal to or greater than 504 sectors.

There are 4 options that can be used with this command: MAXAVAIL,IO, SUMMARY and a user entered sector size replacing the >504 column in the report.

Sometimes it is necessary to find the largest available area on a family; SUPERVISOR can report this by using the MAXAVAIL modifier:

```
FP DEV : Maxavail
```

Unit	Familyname	#x	DiskSize	TotAvail	%	MaxAvail	% Open	Err
501	DEV	#1	17476277	15106011	86	5933736	34	29 0

The IO modifier reports additional I/O transactions data

```
FP D= : IO
```

Unit	Familyname	#x	DiskSize	TotAvail	%	Reads	Writes	Mbyte	Er
501	DEV	#1	17476277	15106011	86	357570	291825	13191	
506	DEVAUDIT	#1	5915	1383	23	26	1	1	
504	DEVDLPACK	#1	5915	1887	32	26	1	1	
500	DISK	#1	17476277	6134824	35	588617	927016	6402	
508	DLPACK	#1	5915	1887	32	26	1	1	
503	DRPACK	#1	5915	1887	32	27	1	1	
** TOTALS **			34976214	21247879	61	946292	1218845	19598	

If the SUMMARY modifier is entered, a report will be generated showing only a summary line for each disk family. See that TEST5 is the accumulated figures for three units.

FP:Summary									
Unit	Familyname	#x	DiskSize	TotAvail	%	>504	% Open	Err	
507	CDIMAGE	#1	34952554	25675366	73	25675366	73	0	0
501	DEV	#1	17476277	15106011	86	15105153	86	29	0
506	DEVAUDIT	#1	5915	1383	23	1383	23	0	0
504	DEVDLPACK	#1	5915	1887	32	1887	32	0	0
500	DISK	#1	17476277	6134824	35	6019183	34	236	0
508	DLPACK	#1	5915	1887	32	1887	32	0	0
503	DRPACK	#1	5915	1887	32	1887	32	0	0
505	LINC17	#1	5915	207	3	0	0	0	0
	TEST5		17745	13661	77	5887	33	0	0
** TOTALS **			69952428	46937113	67	25675366	37	265	0

The following will report the number of sectors available in areas that are equal to or greater than 10,000 sectors within pack families whose ids end in the string "PACK":

```
FP =PACK: > 10000
```

Unit	Familyname	#x	DiskSize	TotAvail	%	>10000	% Open	Err
504	DEVDLPACK	#1	5915	1887	32	0	0	0
508	DLPACK	#1	5915	1887	32	0	0	0
503	DRPACK	#1	5915	1887	32	0	0	0
** TOTALS **			17745	5661	32	0	0	0

Headers Open (HO) Command

The HO (Headers Open) command reports on disk files that are open on a family including temporary and job files. The list of returned headers may be filtered by the inclusion of a family index in the command. This command can be useful in determining if a program goes into an infinite loop writing to a temporary file. Programs using the pack that contains this file will start to get SECTORS REQUIRED RSVPs. This command can also be used to identify what actions are needed to close a pack.

A simple example of HO DISK :

```
TT +
-Tmp--Users---Size----- Name -----
T      1      540  *BUTTERFLY
      1      900  *COMS/CF ILE
      1     27648  *COMS/INPQ
      1     1512  *COMS/TTRAIL/TPLIBRARY/0001
      1      504  *CONF IGURATION/NETWORKSUPPORT/DSS/ATTRIBUTES
      1      504  *CONF IGURATION/NETWORKSUPPORT/DSS/ENTITIES
      1      144  *CONF IGURATION/NETWORKSUPPORT/MINIMALNAME
      1       30  *CONF IGURATION/TELNETSUPPORT/VALUES
      2     1512  *DMSUPPORT/METATAPELIB4
      1     2000  *HLCN/TERMINIT/INFO
      2     1350  *JOBDESC
      1      300  *KEYEDIOII/CONTROL
      1     1000  *MDPF/ASSISTANT/STATE
      1     20916  *METALOGIC/COPYWRITE
      4     3294  *METALOGIC/COPYWRITE/BRIDGE/TCP IP
      3     3852  *METALOGIC/COPYWRITE/HOTLINEAGENT
      4     1062  *METALOGIC/DBCONTROL
      2     7920  *METALOGIC/FLEX/LIBRARY
      2     5130  *METALOGIC/HTTP
      2     1674  *METALOGIC/MAGUS
      2     2484  *METALOGIC/MAILLIB
Window S/1 at COURSE2MCP
```

Each line of the response is preceded by: "T" for temporary, " " for permanent, "J" for job. This marker is followed by the open count of the file, its size in sectors and the file title.

By putting a prefix of PR before the HO command, the output is sent to a printer backup file. The file is closed immediately upon completion of the report and an appropriate message "REPORT PRINTED BY JOB <jobnumber>" is generated.

```
TT PR HO DBPACK
```

To see just those files matching a wildcard pattern, add a pattern after the family.

```
HO DISK =META=
-Tmp--Users---Size----- Name -----
2      1512    *DMSUPPORT/METATAPELIB4
4      3294    *METALOGIC/COPYWRITE/BRIDGE/TCP IP
3      3852    *METALOGIC/COPYWRITE/HOTLINEAGENT
1     20916    *METALOGIC/COPYWRITE
4      1062    *METALOGIC/DBCONTROL
2      7920    *METALOGIC/FLEX/LIBRARY
2      5130    *METALOGIC/HTTP
2      1674    *METALOGIC/MAGUS
1      2000    *METALOGIC/MAILLIB/CONFIG
1      2040    *METALOGIC/MAILLIB/LOG
2      2484    *METALOGIC/MAILLIB
2      1764    *METALOGIC/OPALTAPELIB4
1      5100    *METALOGIC/SUPERVISOR/LOG
1      2556    *METALOGIC/SUPERVISOR/NAPINTERFACE
2        990    *METALOGIC/SUPERVISOR/RECORDER
1        432    *METALOGIC/SUPERVISOR/WAITWATCHER
6     16884    *METALOGIC/SUPERVISOR
1        510    *METALOGIC/TRIM/LOG
1     15000    *METATAPELIB4/AUDIT1
```

Window S/1 at COURSE2MCP

To see the tasks using an individual file, merely put the full file name.

```
HO DISK METALOGIC/MAGUS
```

```
----- File is in use by 2 Users -----
08777\08777 *METALOGIC/MAGUS
SL Library
```

Files opened by an application with EXCLUSIVE=TRUE are flagged. From a normal HO interrogation, such file entries appear with 'x' following the user count:

```
HO DEV SCHED=
-Tmp--Users---Size----- Name -----
1x     57600    *SCHEDULE
```

Or for an individual file:

```
HO DEV SCHEDULE
----- File is in *Exclusive* use by 1 User -----
35707\35707 *METALOGIC/SUPERVISOR
```

Linked Users (LU) Command

The LU (Linked Users) command shows the tasks which are linked to a Library or Database Stack. The syntax of this command is

```
TT LU <mix number list>
```

The mix number list must contain at least one or more mix numbers of Libraries or Database Stacks.

The system's LIBS command and DBS command are used to find the mix numbers of the Library or Database in question. The LU command will report the mix numbers and names for all the tasks directly using the specified Library or Database Stack.

The response from the system LIBS command will look something like:

```
LIBS
---Mix---Frz---Shr---Usr----- 84 FROZEN LIBRARIES -----
* 8802 Temp Priv 1 SL Job (SUPERVISOR) *SYSTEM/JOBFORMATTER
* 8794 Temp All 2 Job (TAPELIB) *DMSUPPORT/METATAPELIB4
* 8795 Ctrl All 1 Job (TAPELIB) *DMSUPPORT/METATAPELIB4
* 8791 Ctrl All 1 SL Job *METALOGIC/MAILLIB
* 8790 Temp All 2 SL Job *METALOGIC/OPALTAPELIB4
* 8781 Ctrl All 0 Job *METALOGIC/DBCONTROL
* 8780 Ctrl All 0 Job (SUPERVISOR) *METALOGIC/SUPERVISOR
* 8777 Perm All 9 SL Job *METALOGIC/MAGUS
* 8776 Ctrl All 0 SL Job *METALOGIC/SUPERVISOR/RECORDER
* 8773 Conn Priv Job *METALOGIC/COPYWRITE/BRIDGE/TCPIP
* 9421 Conn Priv Job (BOB) *METALOGIC/COPYWRITE/HOTLINEAGENT
* 8985 Perm All 0 SL Job *METALOGIC/FLEX/LIBRARY
* 2653 Perm All 0 SL Job *SYSTEM/SLICESUPPORT
* 2643 Temp All 1 SL Job *SYSTEM/FTAMSUPPORT
* 2642 Ctrl All 0 SL Job *SYSTEM/ASSISTANT/TRAPSUPPORT
* 2640 Ctrl All 1 SL Job *SYSTEM/DSSSUPPORT
* 2639 Perm All 1 SL Job *SYSTEM/EVASUPPORT
* 2632 Ctrl All 0 SL Job *SYSTEM/CNSOBJMGR
* 2626 Ctrl All 0 SL Job *SYSTEM/SNMPAGENT
* 2608 Temp All 1 SL Job *SYSTEM/HELP
* 2600 Temp Priv 1 Job *SYSTEM/MARC/AGENDA/TDXXX
Window S/1 at COURSE2MCP
```

If we want to know which nine programs are currently using the support library "METALOGIC/MAGUS", the command "TT LU 8777" should be submitted to SUPERVISOR.

SUPERVISOR's response would be:

```
LU 8777
----- LINKED USERS -----
----- MIX NUMBER 08777 IS LINKED TO 9 STACKS -----
08781/08793 *METALOGIC/DBCONTROL/METATAPELIB4
08791 JOB *METALOGIC/MAILLIB
08790 JOB *METALOGIC/OPALTAPELIB4
08786 JOB *METALOGIC/SUPERVISOR/WAITWATCHER
08781/08785 *METALOGIC/DBCONTROL/IPP/YORK
08781/08784 *METALOGIC/DBCONTROL/IPP/STIRLING
08781 JOB *METALOGIC/DBCONTROL
08780 JOB *METALOGIC/SUPERVISOR
08985 JOB *METALOGIC/FLEX/LIBRARY
```

Tape Directory (PDT) Command

The PDT command invokes SUPERVISOR's fast tape directory routine. PDT has the ability to read a library maintenance tape, the header of a memory dump tape and the keys information on a Unisys keyed tape. It can also handle library maintenance CD-ROMs and LibmaintDirs. PDT is an improved, much faster TAPEDIR to the ODT, replacing the TDIR SPO command. PDT only holds a library tape open for as long as is necessary to read the directory, then closes it. PDT can be executed from any SUPERVISOR terminal, such as the COMS window or the MARC DIRECTIVE.

If PDT finds a memory dump tape, it will list the dump header, with the cause of the dump, when taken, etc.

A typical response to the PDT command "TT PDT MT 80" is as follows:

```
TT +
----- SUPERVISOR PDT FOR MT 80 FLEXCOPY07100AG/FILE000 [HP3011] OF 10/04/2007 --
00001 *NP/SUPPORT/RECOVERY ON DISK
00002 *DBC/TAPELIB/METATAPELIB4/STATISTICS/FRIDAY ON DISK
00003 *DBC/TAPELIB/METATAPELIB4/STATISTICS/MONDAY ON DISK
00004 *DBC/TAPELIB/METATAPELIB4/STATISTICS/TUESDAY ON DISK
00005 *DBC/TAPELIB/METATAPELIB4/STATISTICS/THURSDAY ON DISK
00006 *DBC/TAPELIB/METATAPELIB4/STATISTICS/WEDNESDAY ON DISK
00007 *COMS/CFILE ON DISK
00008 *COMS/TTRAIL/TPLIBRARY/0002 ON DISK
00009 *HLCN/TERMINIT/INFO ON DISK
00010 *TIME/CONFIG ON DISK
00011 *TLOG ON DISK
00012 *SYSTEM/COMS/MARC/COMMANDER/DIALOGINFO ON DISK
00013 *SYSTEM/COMS/MARC/COMMANDER/SESSIONINFO ON DISK
00014 *SYSTEM/PRINT/ONTSUPPORT/DATA/AUDIT1 ON DISK
00015 *SYSTEM/NXSERVICES/CONFIG ON DISK
00016 *SYSTEM/NXSERVICES/NEWCONFIG/ADMINCENTER ON DISK
00017 *SYSTEM/NXSERVICES/CONFIGERRORS/ADMINCENTER ON DISK
00018 *SYSTEM/PRINTERINFO ON DISK
00019 *SYSTEM/SECURITYLOG ON DISK
00020 *SYSTEM/USERDATAFILE ON DISK
00021 *SYSTEM/TCPIPSECURITY/RULES ON DISK
```

Generally this command will produce a multi-page response from SUPERVISOR. When any response has more than 1 page of output SUPERVISOR inserts the next screen command in preparation for the next screen request. To stop displaying the tape directory simply enter your next command. To continue displaying the tape directory, transmit the "TT NS" (or "TT +" on a Window Station) for the next screen until you want to stop or the end of the directory is reached.

```
PDT MT 80
----- SUPERVISOR PDT FOR MT 80 FLEXCOPY07100AG/FILE000 [HP3011] OF 10/04/2007 --
00043 *METALOGIC/FLEX/RULES/CATALOG/STATISTICS ON DEV
00044 *METALOGIC/MAGUS/CONFIGURATIONDATA/DISK ON DEV
00045 (BOB)TEST/MAKE/FILES ON DEV
00046 (BOB)CONFIG/METAMCPB ON DEV
00047 (BOB)OBJECT/TEST/MAKE/FILES ON DEV
00048 (TAPELIB)METATAPELIB4/AUDIT1 ON DEV
00049 (META)SYMBOL/SCHEDULE ON DEV
00050 *SAVED/SCHEDULE ON DEV
00051 (META)BATCH/"MAGUSGEN.CON" ON DEV
00052 *SYSTEM/CATALOG/090407 ON DEV
```

It is also possible to narrow the scope of the PDT command by giving a pattern to search for. It is also possible to scan LibmaintDir files.

PDT on Libmaintdir files currently requires the Metalogic CopyWrite product to be installed. This restriction may be relaxed in a future release.

Example:

```
PDT IN (FLEX)LIBMAINTDIR/DEV_A/20080319/DF0008 ON DISK =SUPERVISOR ON DEV
----- SUPERVISOR PDT IN (FLEX)LIBMAINTDIR/DEV_A/20080319/DF0008 ON DISK -----
----- USING PATTERN =SUPERVISOR ON DEV -----
00016 *OBJECT/SUPERVISOR ON DEV
00017 *METALOGIC/SUPERVISOR ON DEV
00018 (BOB)OBJECT/SUPERVISOR ON DEV
00019 (BOB)METALOGIC/SUPERVISOR ON DEV
```

PDT has other tricks as well. To send the output to the printer, merely precede the command by PR as with the HO and LU commands:

```
PR PDT MT 33
```

Scheduling

Conventional system commands are always executed immediately by the MCP's CONTROLLER. This requires the presence of an operator at those times commands need to be input. However, the Scheduler in SUPERVISOR introduces the ability to delay or schedule system and SUPERVISOR commands for a specific time or times in the future. This means, for example, that jobs can be scheduled to run when the system is unattended, or standard procedures can be invoked in the case of events such as a SUPERVISOR restart or a system HALT/LOAD.

All scheduled activities are held in a file titled *SCHEDULE. The "schedule" may be thought of as an 8 by 1444 matrix, where each element or "bucket" contains the information required by SUPERVISOR to perform one or more scheduled activities.

The eight rows represent each day of the week plus one for today. The 1444 columns represent 1440 minutes in a day plus four extra buckets for HALTLOAD, RESTART, SHIFT BEGIN and SHIFT END.

At midnight each day, SUPERVISOR copies all slots from the new day of the week to the row for TODAY. All prior information in the TODAY row is discarded. The AFTER command is used to access this simple, but powerful time scheduler

Date formats

SUPERVISOR can use two different date formats: USDATES in the form MM/DD/YYYY or the non-American format of DD/MM/YYYY. You should check to see what your system is using through the SO command:

```
SO
----- SUPERVISOR OPTIONS -----
1 HARDCOPY          2 USERNOFOR
3 LOGMINIMAL        4 NOAUTODC
5*WAITOKTASK        7 NODEFACTIVE
8 STRICTODTS        10 MONITORING
13 LOGOPERATOR      18 NAMEOPERATORS
19 USDATES           22*TAPELIB
28 NOTLOGGING       30 DEFINERESP
31 ASYNCWFL         32 NOWHENRESTART
33 NOVALIDATE       35*AUTOFS
36 NOSTATS          37*NOENTERLIST
38 AUTOTL           45 USECOPYWRITE
46 COMPRESSDEFINE   47*LOGGING
```

Options that have been set have an asterisk (*) beside them. If you want to use the USDATES format, you should set option 19:

```
TT SO 19
```

Unless otherwise stated, in the examples that follow it is assumed that USDATES is reset and the non-American (DD/MM/YYYY) format is used.

Within SUPERVISOR itself, all date-handling functionality contained within the OPAL compiler and run-time machine will correctly handle Year 2000. SUPERVISOR's scheduling functionality has also been enhanced for dates before and after 2000. Various reports generated by these two packages have also been enhanced to include 4-digit years.

Within the OPAL engine, date and time intrinsic handling supports both 2- and 4-digit years. Where possible, OPAL date attributes have been changed to use 4-digit years, particularly for files, though the older 2-digit versions are still available for backward compatibility. Where applicable, and dependent on MCP version, the general Metalogic approach has been to treat two digit dates less than 70 as being in this century (2000), and dates greater than or equal to 70 as being in the previous century (1900).

AFTER command

The [AFTER](#) command is used for adding, deleting or interrogating an activity in the schedule. The scheduled activity will be performed as soon as possible after the scheduled time.

AFTER can be used to schedule operator commands, WFL jobs or other Supervisor activities at any time of day, on a daily, weekly or monthly basis or many other possible combinations. Other modifiers available with AFTER include HALTLOAD, RESTART or even COLDSTART.

The syntax is

```
AFTER <modifier> <time qualification> <text>
```

The modifier may be "+" (plus sign), " " (blank), "-" (minus sign), or "?" (question mark). <text> in this command is something which would be entered at the ODT such as SUPERVISOR commands, system commands, OPAL programs executed using EVALUATE, WHEN, ONCE or DO, etc.

The AFTER command with a modifier of "+" implies that SUPERVISOR should schedule the <text> for each of the specified times if and only if <text> is not already scheduled at that time.

If a modifier of " " (blank) is used, the activity is scheduled unconditionally. If a modifier of "-" (minus) is used, SUPERVISOR will delete the selected activity, scheduled at each of the specified times.

AFTER TRIGGER

Trigger is followed by an identifier with an optional usercode.

Ex.

```
FOR LIVE/PW AF TRIGGER (BOB)TEST DAILY:START BOB/TEST
```

Once this is in the schedule, entering

```
TT TRIGGER (BOB)TEST
```


will cause BOB/TEST to be started under the usercode LIVE. The intent of this change is to allow jobs to be stored in the schedule with appropriate Usecode and accesscode and then be 'Triggered' by a subsequent event, perhaps the successful completion of another Job. It also allows an operator to start a job under a specific usercode without needing access to the password for that user.

AFTER date and time modifiers

Time is specified in military time format. The first minute of a day is 0000 – midnight – and the last is 2359 – one minute to midnight. For example:

AFTER 2315

specifies that the activity should be scheduled for 11:15PM for TODAY and only TODAY. If the ON qualification is not used, the default of TODAY is assumed. Remember that TODAY's schedule will be overwritten at midnight. Lists of times are permitted as well as periodic intervals. If you want to schedule an activity for more than one occurrence, you must say so.

AFTER 0900, 1030, 1400

specifies that the activity should be scheduled at 9:00AM, 10:30AM and 2:00PM. Again, this will only apply to TODAY.

AFTER 0900-1000 (10)

specifies the activity should be scheduled for every 10 minutes between the hours of 9:00AM and 10:00AM. An AFTER command also may contain certain special "minutes", designated by key words such as HL (HaltLoad).

A colon is allowed in the time, so AF 10:20 is also valid.

It has a potential problem in very unlikely cases.

If an attempt is made to schedule the command 59OK at a time before 00:12 then the syntax AF 12:59OK will schedule the command OK at 12:59 For the desired command to work enter 0012:59OK or 00:12:59 OK

Time can be further specified by day of the week. The simplest cases are DAILY, TODAY, TOMORROW, and ON DATE (used for defining specific dates, format based on SUPERVISOR's option USDATES). For example:

AF 1300 DAILY ...

AF 2315 TOMORROW ...

AF 2355 ON DATE 31/12/2008 ..

In the first example, the activity will be scheduled for 1:00PM DAILY means all days of the week. In the third example the date entered is in the default format (with SUPERVISOR option USDATES reset).

SUPERVISOR also is capable of scheduling by days of the week. In addition to DAILY, WEEKDAYS means Monday through Friday, WEEKENDS means Saturday and Sunday, while WORKDAYS means WEEKDAYS except HOLIDAYS.

For example:

```
AF 2325 ON MONDAY, WEEKENDS ...  
AF HL ON WORKDAYS ....
```

Holidays are DEFINEd by using the HOLIDAY command. The WORKDAYS command may be used to specify certain Saturday or Sundays as additional days to behave as normal working days. Both commands are covered in more detail later in this section.

Another common requirement is for specific days in the month. The day can thus be qualified by an ordinal number, or the keywords LAST and NEXT <day of week>

```
AF 900 ON FIRST MONDAY ...  
AF 1800 ON LAST WORKDAYS...  
AF 1030 ON NEXT TUESDAY ...
```

Lastly, activities may be restricted to specific months of the year by the means of an OF qualification. Months are referenced by their English names and list and ranges are permitted. For example:

```
AF HL DAILY OF JANUARY ...  
AF HL DAILY OF FEBRUARY, JUNE ...  
AF HL DAILY OF APRIL-JULY, SEPTEMBER ...
```

In the first example, an activity will be scheduled every day of January. The second example will schedule an activity for every day of February and June. The third example will schedule an activity every day of April, May, June, July and September.

The WHERE modifier allows for user programmable restrictions. WHERE is followed by the name of a system situation. If all other conditions for the after are true then the situation is evaluated. If it returns true the AFTER will run.

```
Ex. AF 1000 ON WORKDAYS WHERE Hol_Yesterday START MY/JOB
```

If Hol_Yesterday is defined as:

```
TT DEFINE + SITUATION Hol_YESTERDAY:  
Holiday(NewdateTStoday,-1))
```

Then the above AF would only run on Workdays where yesterday was a holiday.

Using the AFTER command

Remembering that the syntax of the AFTER command is

```
AFTER <modifier>:<time specification> <text>
```

Text following the time specification is normally a command that could be entered at an ODT.

For example to set the MCP option NODUMP at 11:00PM today enter:

```
AF 2300:OP + NODUMP
```


If the "+" (plus sign) is used as a modifier, the activity will not be scheduled for a time where it is already present. Valid modifiers are " " (blank) for unconditional adds, "+" (plus sign) for conditional adds, "-" (minus sign) for deletes, and "?" (question mark) for inquiries about the schedule.

SUPERVISOR also has the capability to perform lookup functions in scheduled activities. When a scheduled activity is about to be performed, and it is not a SUPERVISOR command, the text is re-examined to evaluate any lookup functions that may be present. This lookup process allows many commands, which could not otherwise be scheduled, to be specified in advance without exact knowledge of the parameters necessary in a valid system command.

For example, if we want to schedule SYSTEM/CANDE to quit at 9:00PM daily, the following command should be entered into the schedule:

```
AF 2100 DAILY:#[MX=SYSTEM/CANDE] SM QUIT
```

Lookup functions may start with the keywords MX or PK or may be a character mnemonic. The value returned from an MX Lookup is a mix number list, from PK it is a unit number list. A character mnemonic might be used to highlight, underline or blink a message to an operator.

To schedule *SYSTEM/CANDE for execution every day at 8:00AM in SUPERVISOR's Scheduler, enter

```
AF 0800 DAILY: ??RUN *SYSTEM/CANDE
```

SUPERVISOR will respond with

```
SCHEDULED IN 7 SUNDAY-SATURDAY SLOTS  
NOT SCHEDULED IN 1 TODAY SLOT:PAST TIME(S) IGNORED
```

The response from SUPERVISOR indicates that *SYSTEM/CANDE was placed in the schedule for Sunday through Saturday but the time slot for TODAY was ignored, since it was after 8:00AM when SUPERVISOR attempted to place this entry in the schedule. To query the schedule enter

```
AF ? <time specification>
```

To see all days, Sunday through Saturday, for all entries between midnight and 11:59PM, enter

```
AF ? 0-2359 DAILY
----- PENDING SCHEDULE AFTER 0800 ON SUNDAY -----
1:?? RUN *SYSTEM/CANDE
2:TT SAVE SCHEDULE AS *SAVED/SCHEDULE/TEXT

----- PENDING SCHEDULE AFTER 0800 ON MONDAY -----
1:?? RUN *SYSTEM/CANDE

----- PENDING SCHEDULE AFTER 0800 ON TUESDAY -----
1:?? RUN *SYSTEM/CANDE
2:TT DO CONDITIONALSA - VE

----- PENDING SCHEDULE AFTER 0800 ON WEDNESDAY -----
1:?? RUN *SYSTEM/CANDE

----- PENDING SCHEDULE AFTER 0800 ON THURSDAY -----
1: ?? RUN *SYSTEM/CANDE

----- PENDING SCHEDULE AFTER 0800 ON FRIDAY -----
1:?? RUN *SYSTEM/CANDE

----- PENDING SCHEDULE AFTER 0800 ON SATURDAY -----
1:?? RUN *SYSTEM/CANDE
```

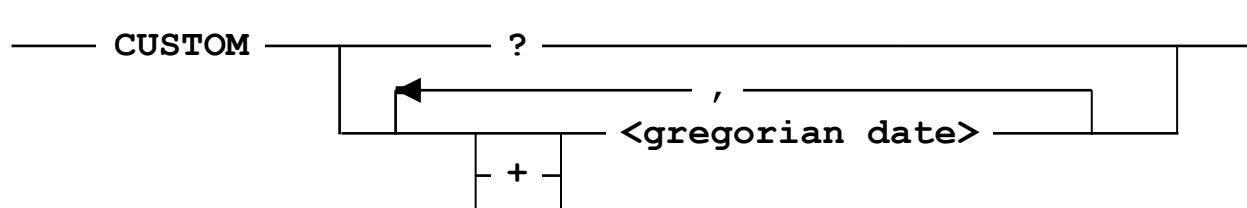
To delete the run of SYSTEM/CANDE from each day with one command:

```
AF - 0800 DAILY : ?? RUN *SYSTEM/CANDE
DELETED FROM 7 SUNDAY-SATURDAY SLOTS
NOT DELETED FROM 1 TODAY SLOT:PAST TIME(S) IGNORED
```

The second way is to enter the activity number. The following example will delete the first activity scheduled for today at 11:00PM. SUPERVISOR will response immediately with "DELETING : <text>" where <text> is the activity from the schedule being deleted.

```
AF - 2300 @1
DELETED FROM ONE TODAY SLOT
```

CUSTOM Command



Similar to WORKDAYS and HOLIDAYS, CUSTOM allows multiple special dates to be assigned for regular site usage throughout the year. These dates are available for

AFTER scheduling activities and OPAL scripting.

HOLIDAY command

The HOLIDAY (or HOLIDAYS) command adds or deletes dates from the list of holidays or displays the current list of holidays. There are two types of holidays. The first type is holidays where the date of the holiday varies within each year (such as Easter). These movable holidays must be DEFINEd by giving the date for each year. The second type is holidays that always occur on the same day of the same month (like American Independence Day or Christmas). These fixed holidays may be DEFINEd by entering only the day and the month.

In the example that follows the SUPERVISOR option USDATES is set so the format of the date is input as MM/DD/YYYY. To add Christmas Eve, Christmas Day, Good Friday 1998 and American Independence Day enter:

```
TT HOLIDAY + 12/24, 12/25, 4/1/2008, 4/7
```

SUPERVISOR will respond with:

```
----- HOLIDAYS -----  
APRIL          1 ,2008  
JULY           4  
DECEMBER       24  
DECEMBER       25
```

To view the current list of holidays on your system enter

```
TT HOLIDAYS ?
```

and SUPERVISOR will display the list of holidays held in your *SCHEDULE file.

[Full Description.](#)

WORKDAYS command

The WORKDAYS command allows a site to specify certain Saturday or Sundays as additional working days. Any such dates will be considered for activities scheduled by any variant of the AFTER .. WORKDAYS command subset. The WORKDAYS command is similar to HOLIDAYS and has identical syntax.

All reference to activities using the clause 'ON WORKDAYS' will include any additional weekend working days that have been assigned by the WORKDAYS command. Only dates that map onto a Saturday or a Sunday can be assigned by the WORKDAYS command; other days will generate a syntax error.

Also, Supervisor will automatically add the current year if a date is given with only day and month values. The '+' is not required for adding new WORKDAYS, but is allowed for both the HOLIDAYS and WORKDAYS commands.

Some examples:

```
TT WORKDAYS + 21/4/02, 8/6, 17/8
TT WORKDAYS + 4/1/2003
----- WORKDAYS -----
      JANUARY           4, 2003
```

For most sites, the WORKDAYS command is not applicable; however, for those countries where Saturday and Sunday can be regarded as optional working days, this may be a useful feature.

[Full Description.](#)

OPAL Programming

OPAL (**OP**erations **A**lgorithmic **L**anguage) is a problem-oriented language designed for solving the problem of writing operational software for Unisys A Series systems. It is a simple, yet powerful language based on Unisys ALGOL, WFL (Work Flow Language) and system commands. Programmers with experience in ALGOL or WFL will quickly recognise its similarities and appreciate the power of its extensions to those languages.

An important difference between OPAL and WFL is the use of an object-oriented paradigm. OPAL views its environment as consisting of different classes of objects. OPAL has the means to search for objects of a given object class, report on them, and manipulate them.

Metalogic has DEFINEd many object classes in the MCP universe, and they are documented within OPAL.

The available object classes can be seen using the HELP CONTEXT command (as of Supervisor version 570.17):

TT HELP CONTEXTS

```
TT +
----- CONTEXTS -----
AFTER = SUNDAY, MONDAY, TUESDAY, WEDNESDAY, THURSDAY, FRIDAY, SATURDAY,
        COLDSTART, TODAY
C(COMPLETED)
CUSTOM
DATABASE = OPEN, CLOSE
DBACCESS
DEFINE = SITU, ODTS, DISP, MEMO, COMMAND
DMS = START, STOP
EI
FILECLOSE = CLOSE, PLI
FILEOPEN
FILESTATUS = CREATIONCOPY, CREATION, REMOVAL, TITLE(TITLECHANGE), SECURITY,
            COPY, DESTROY, RELEASE
HTTP = <INTEGER>
JOB
JOBREJECT
SQ(JOBQUEUE) = <INTEGER>
LOG = <INTEGER>, <INTEGER LIST>
LOGBOJ = BOJ, BOT
LOGEOJ = EOJ, EOT
LOGHTTP
LOGOFF
LOGON
LOGPS = CREATED, COMPLETION, STARTED, PRINTED
MAIL
MCSSECURITY
METALOG = SUPERVISOR(SUP), TRIM, MAIL(MAILLIB), SITE, FLEX
MSG(MESSAGE)
MX(MIX) = A(ACTIVE), LIBS(LIBRARIES), S(SCHEDULED), W(WAITING), GOING, PR,
        (PRIORITY), BOT(BOJ), DBS
NAPLOG = <INTEGER>
OPERATOR = SETSTATUS, CONTROLLER, PS, PRIMITIVE
PD
PER = MT, LP(TP), DK, SC(ODT), CD(CDROM), TSP, HY, CR, CP, DC, PK, UD, HC, CDR,
        SPC, ARP, IP, NP, VSID, LBP, VC
PRINTS
SECURITY
SESSIONS
SHOWOPEN = DK, PK, CD
SL
SMTP
STATIONS = NIF, PSEUDO
SYSTEM
TAPEDB = PENDING(BYPENDING), BYNAME(NAME), BYFAMILY(FAMILY), BYLOCATION,
```

Continuation screen.

```
HELP CONTEXTS
----- CONTEXTS -----
        (LOCATION), SCRATCH(BYSCRATCH), BYTS(TS), BYPOOL(POOL), BYVOL
TAPELABEL
U(USER)
VDBS
VL = TAPE, DISK, PACK, CD, CDIMAGE(CDR), FARM(REMOTE)
VOLUME = ONLINE, OFFLINE, TAPEPG, TAPEEXPIRY, TAPENew, TAPEUSE, TAPEHOLD
WHEN
WINDOWS = REMOTE, MCS, DIRECT
```

Each line is an available object class. The object classes followed by an '=', such as MX, have subclasses. Subclasses can be used when defining Situations and tell it to limit its search, e.g. MX=W, specifies that a SITUation should not search for all

mix entries, just those which are waiting. The object classes such as MSG, or subclasses like W, with ids in parentheses, specify alternate names, thus MESSAGE is accepted as a synonym of MSG, WAITING as a synonym for W.

Each object class is DEFINEd by its attribute subset. Attributes are "pieces" of information about a specific component of the system or its program environment. Each piece of information is given a key word. Attributes provide access to almost all the information about the system. Another variant of TT HELP, allows us to see all of the attributes of an object class:

```
TT HELP ATT =: <object class>
```

For example, to retrieve information about all MSG attributes:

```
HELP ATT =:MSG
----- HELP ATTRIBUTES -----
CODEMIXNO      MESSAGE
IDENTITY       MESSAGE
IMPORTED       MESSAGE
JOBNO          MESSAGE
JOBNUMBER      MESSAGE
MIXNO          MESSAGE
MIXNUMBER      MESSAGE
MSGCAT         MESSAGE
MSGDATE        MESSAGE
MSGDAY         MESSAGE
MSGIDENTITY    MESSAGE
MSGNO          MESSAGE
MSGNUMBER      MESSAGE
MSGPARAM       MESSAGE
MSGTIME        MESSAGE
MSGTIMESTAMP   MESSAGE
MSGTS          MESSAGE
MSGTYPE        MESSAGE
SESSIONNO      MESSAGE
SESSIONNUMBER  MESSAGE
TEXT           MESSAGE
Window S/1 at DELL MCP
```

Or, specifying a more detailed wildcard string with a context:

```
HELP ATT =TIME:LOGE0J
----- HELP ATTRIBUTES -----
ACCUMIOTIME    LOGE0J
ACCUMPROCTIME  LOGE0J
ELAPSED TIME   LOGE0J
HOLDQTIME      LOGE0J
IMPACTTIME     LOGE0J
INITIALPBITTIME LOGE0J
IOTIME         LOGE0J
LOGENDTIME     LOG
LOGSTARTTIME   LOG
LOGTIME        LOG
MEASUREDINITPBITTIME LOGE0J
MEASUREDOTHERPBITTIME LOGE0J
MEASUREDPROCTIME LOGE0J
NORMINITPBITTIME LOGE0J
NORMOTHERPBITTIME LOGE0J
OTHERPBITTIME  LOGE0J
PROCTIME       LOGE0J
READYTIME      LOGE0J
STARTTIME      LOGE0J
```

MSG is one of the smallest object classes - some classes have hundreds of attributes. If the object class is not specified, synonyms from all object classes are returned, the object class being given after the attribute name. The '=' is actually a pattern, which is useful for browsing around the attributes.

For example, to find out which attributes contain the string "IDENT":

```
HELP ATT =IDENT=+
---- HELP ATTRIBUTES ----
IDENTITY          MIX
IDENTITY          MESSAGE
IDENTITY          PD
IDENTITY          USER
IG_IDENTITY       SYSTEM
MSGIDENTITY       MESSAGE
NORESIDENTRIES    PD
PDIDENTITY        SYSTEM PD
RESIDENT          PD
RESIDENTENTRY     PD
RESIDENTLIMIT     VDBS
RESIDENTPROGRAM   LOGBOJ
RESIDENTPROGRAM   LOGEOJ
RGIDENTIFIER      SYSTEM NAP
TOTALRESIDENT     VDBS
```

The Help Attributes command when passed a pattern, lists the attributes in two columns. Depending on the length of the longest attribute name to be displayed and the width of the screen, some data may need to be truncated to fit. If either an entry in the first column or the second column would be truncated, only the entry in the first column is shown. The entry which would have been in the second column is moved to the next line.

To get detailed information about a particular attribute, say IDENTITY:

```
HELP ATT IDENTITY
---- HELP ATTRIBUTES ----
IDENTITY (MIX) Returns STRING in the form of an identifier
Semantics : The string that is used to replace the text 'DISPLAY' in
a DISPLAY message.

IDENTITY (MESSAGE) Returns STRING
MSGIDENTITY is a synonym for this attribute
Semantics : returns the IDENTITY of the task that issued the message,
as established by the MP <file> + IDENTITY command.

IDENTITY (PD) Returns STRING in the form of a file title
Semantics : Gives the IDENTITY as set by the MP IDENTITY ODT command

IDENTITY (USER) Returns STRING
Semantics : Site defined value normally with user's name/address
```

If you make an error in keying in the ATtribute name or enter an ATtribute name that does not exist, the system will respond with "ATTRIBUTE NOT FOUND" below your original input.

Further SYSTEM Attributes in the VL, PD and NAP subsets can be displayed by specifying the subset as follows:

```
TT HELP ATT = : SYSTEM NAP
TT HELP ATT = :SYSTEM ALL
will show all basic system attributes and all of the subsets
```

To produce a report showing all ATtributes available to an OPAL programmer and a concise description of their meanings, the PRINT ATT command is used. This produces an alphabetical list of ATtributes sorted by context.

PRINT ATT indents each entry by 15 characters if the attribute name length is less than 16 characters. For longer name lengths the entry is indented by one character more than the attribute length.

SUPERVISOR will respond with "PRINT COMPLETED" when the report has been generated.

This will create a very large print file

Object Classes

The following is a full list of all the OPAL program contexts currently available within SUPERVISOR.

Context	Description
AFTER	Supervisor schedule information
COMPLETED	Completed tasks or jobs
CUSTOM	Allows access to user defined contexts
DATABASE	LOG database open and close records
DMS	LOG database FREEZE and RESUME
DBACCESS	DMSII access log entries
DEFINE	Defined Opal programs
EI	Establish identity events
FILECLOSE	LOG file close records
FILEOPEN	LOG file open records
FILESTATUS	LOG file status records
HTTP	Requests from web browsers.
JOB	Limited job-tracking using ON JOBMESSAGE
JOBQUEUE (SQ)	Capture of queued WFL job information
JOBREJECT	LOG tracking of rejected jobs
LOG	Generic SUMLOG records
LOGBOJ	LOG beginning-of-job records
LOGEOJ	LOG end-of-job records
LOGHTTP	LOG HTTP request
LOGOFF	LOG session log-on records
LOGON	LOG session log-off records
LOGPS	LOG PrintS event records
MAIL	Metalogic Mail program
MCSSECURITY	LOG MCS security records
METALOG	Metalogic log files
MSG	System & program display messages
MX	Task information
NAPLOG	NAPLOG events from the NAP system

Context	Description
OPERATOR	System commands from the ODT
PD	File information
PER	Peripheral information
PRINTS	Print System request information
SECURITY	System security violations
SESSIONS	COMS SESSIONS
SHOWOPEN	Open Files
SL	SLed Libraries
SMTP	Input from mail programs
SQ	System queue access
STATIONS	Datacom stations
SYSTEM	Global system information
TAPEDB	Tapes logged in the Metalogic TRIM system
TAPELABEL	Tape creation
USER	Usercode attributes from USERDATAFILE
VDBS	Database statistics and audit info
VL	Volume Library directory analyser
VOLUME	Volume status changes of tapes, packs, CDs
WHEN	Supervisor 'when' slots
WINDOWS	COMS Windows open and close actions

All OPAL programs can access SYSTEM attributes from any context, but other attributes may only be accessed from a matching context. Thus OPALs with a PER context can only directly access PER and SYSTEM ATTRIBUTES.

Expert System Program Types

Together, SUPERVISOR and OPAL combine to form a powerful knowledge-based, expert system. OPAL works with four types of algorithm: pattern recognition programs called SITUATIONS, which are used to search an object class to find particular objects that satisfy a condition; programs containing system and SUPERVISOR instructions called ODTSequences that act on objects, and report programs called DISPLAYS. COMMANDS are a special form of ODTSequence which can be invoked from Supervisor windows without a DO prefix. SUPERVISOR also has a fifth type, not part of OPAL, called a MEMO, which is used to provide extra on-line help.

These four OPAL algorithm types match the functional areas of an expert system: receiving inputs from its environment; determining actions or responses; and delivering outputs to its operators.

This natural division of behaviour – information input, direct action, and communication – is mirrored in SUPERVISOR. It looks for a given set of circumstances (an OPAL SITUATION) and then responds by communicating (an

OPAL DISPlay) or by direct action (an OPAL ODTSequence or COMMAND). Writing the appropriate SITUations, ODTSequences, COMMANDs and DISPlays to handle any particular circumstance is merely a reflection of our own day-to-day experience.

SITUations

A SITUation is in the form of a Boolean expression that indicates whether an event has occurred or a condition is present. A very simple SITUation to apply to the MX class to determine which tasks have accumulated more than 30 seconds of IO time:

```
DEF SITU TEST (MX) :  
IOTIME > 30
```

where 'IOTIME' is an attribute of the MX object class (look at HELP ATT IOTIME:MX), '>' is the relational operator meaning 'greater than', and '30' is an arithmetic constant.

DISPlays

A DISPlay is a program which solves the problem of writing a report to the terminal or line printer. The syntax is similar to the syntax of the ADM EVENT PRINTLABEL system command.

A simple DISPlay that could be used with our MX SITUation above might look like this:

```
DEF DISP TEST (MX) :  
MIXNO 5 , , TIME (IOTIME) , , NAME
```

where 'MIXNO', 'IOTIME', and 'NAME' are attributes of the MX object class, 'TIME' is a function, which converts a number of seconds into HH:MM:SS format, '5' says to make this field 5 characters wide, ',', is a concatenation operator, which joins 2 strings together and inserts a blank space in between the 2 strings.

The full systax of the DEFINE (DEF) command will be described later as will The EVALUATE(EV) command. But to test this small example enter:

```
EV TEST DISP TEST
```

ODTSequences

An ODTSequence is a sequence of statements, separated by semicolons, which describe an action to be taken. Often it is necessary to group together a sequence of system commands that are understood to be one logical operational function or to simulate an ODT input.

If we wanted to take increase the priority of a mix entry, we could write an ODTSequence like this:

```
DEF ODTs TEST (MX) :  
ODT (MIXNO , "PR" , PRIORITY+1)
```

where 'ODT' is a statement, which passes the string in parentheses to the MCP as an ODT command, 'MIXNO' and 'PRIORITY' are attributes of the MX object class, ',' is a concatenation operator, '+' is an arithmetic operator and '1' is an arithmetic constant.

```
DO TEST 12345
```

Would increase the priority of mix number 12345 by 1.

COMmands

A COMmand uses the same syntax as an ODTSequence for its code. The significant difference is in the way it is executed. From a Supervisor window only the name of the COMmand needs to be entered. This makes it possible to replace standard MCP commands with enhanced Supervisor versions.

The file OPALS/SUPERVISOR/GETTINGSTARTED contains several examples of such COMmands.

A full description of the routines available in the 'GETTINGSTARTED' file are described [in the Advanced Opal section](#).

The DBS COMmand give much more information than the standard MCP version.

It can be loaded into Supervisor by entering:

```
TT ENTER COM DBS FROM *OPALS/SUPERVISOR/GETTINGSTARTED
```

Example of its use from a Supervisor Window

```
dbS
  Mix Users      1 Active Database
  4105          6 (BOB)ACDB1
  Allowed ..... 30000 InUseCore .... 888
  MaxInUse ..... 888   MaxBuff ..... 2
  OverlayGoal .. 5.000% OverlayRate .. 0.000%
  Transactions . 3     ForcedOverlays 0
  NormalAudits . 0     Syncpoints ... 0
  Programs in Transaction state:
    T0004086 U 50 (BOB)OBJECT/AC/DB1/MOD/3 ON DEV
    T0004084 U 50 (BOB)OBJECT/AC/DB1/MOD/2 ON DEV
    T0004081 U 50 (BOB)OBJECT/AC/DB1/MOD/1 ON DEV
  Programs not in Transaction state:
    0004096 I 50 (BOB)OBJECT/AC/DB1/INQ/3 ON DEV
    0004095 I 50 (BOB)OBJECT/AC/DB1/INQ/2 ON DEV
    0004094 I 50 (BOB)OBJECT/AC/DB1/INQ/1 ON DEV
```

The original MCP command is still available with an ODT prefix.

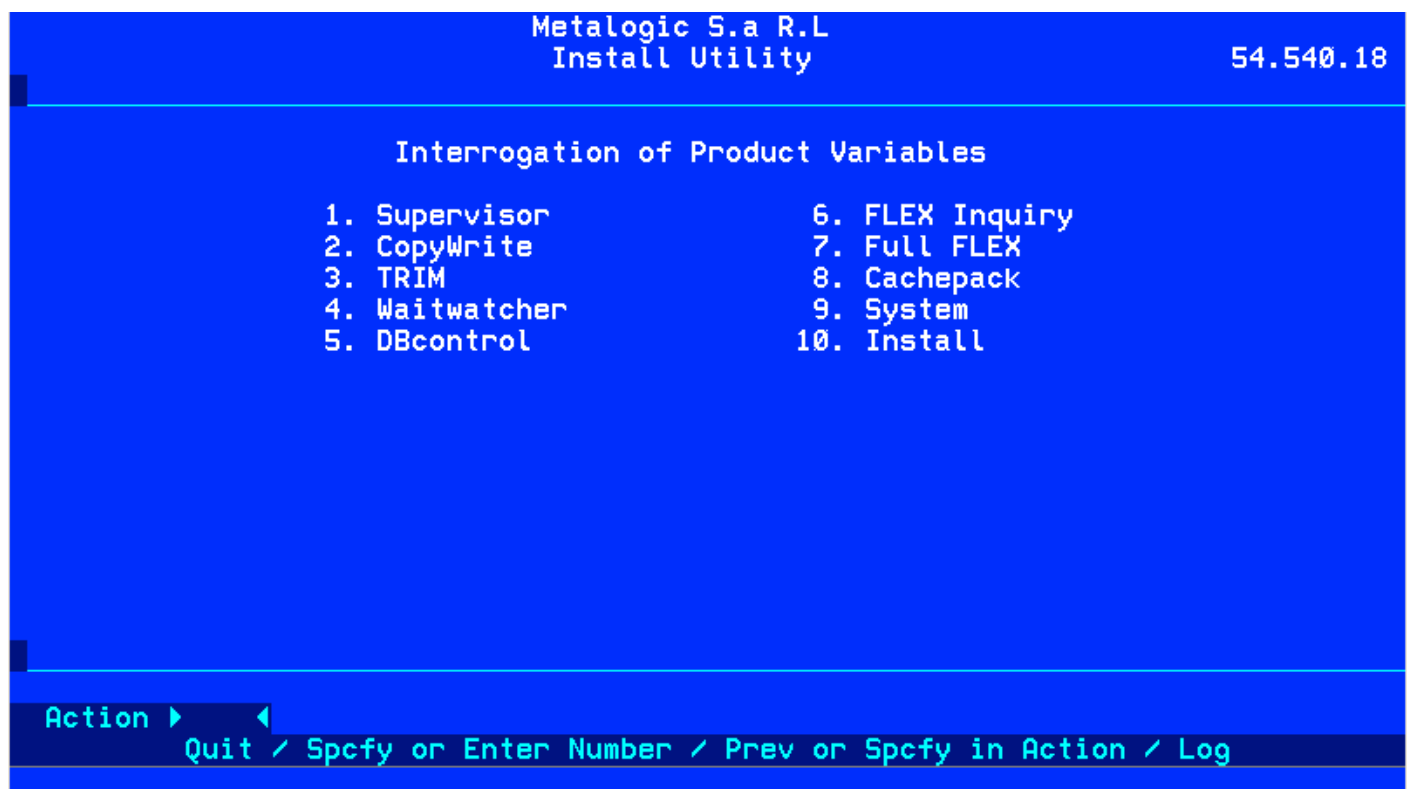
```
ODT DBS
---Mix-Pri--Usr----- 1 ACTIVE DATABASE -----
* 4105 50 6 Job (BOB) (BOB)ACDB1
```

It must be remembered that SUPERVISOR is, in effect, an extension of the operating system and should therefore be used in a responsible fashion. ODTSequences can contain any valid system command and care should be taken when entering these. It is always a good idea to try out new OPAL programs with the TEST option which shows what commands would be executed without actually performing them.

See [Advanced Opal](#) for descriptions of more useful commands.

WHEN SLOTS

Any active Opal program runs in a 'WHEN SLOT' by default there are 40 slots available to run Opal programs. This Limit may be increased to a current maximum of 120, via U *META/INSTALL CONFIG



From the first screen enter 1 for Supervisor options

```
Screen 1 of 2                               Metalogic S.a R.L.
                                           Install Utility
                                           54.540.18
                                           Supervisor Configuration Variables

1. SUP_BACKUPFAM ..... DEV
2. SUP_BACKTRACK ..... NONE
3. SUP_BDNAME ..... No value defined
4. SUP_CLASS ..... 255
5. SUP_DEFCHARGE ..... No value defined
6. SUP_DEFOPALS ..... *SYMBOL/SCHEDULE
7. SUP_DESTINATION ... MAIL:BOBANDIAN@METALOG.COM
8. SUP_EXTERNALFAM ... DEV
9. SUP_EXTERNALUSER .. IPP
10. SUP_LOGBACKUPFAM .. CDIMAGE
11. SUP_MAILUSER ..... No value defined
12. SUP_MAXWHENS ..... No value defined
13. SUP_MAXWINDOWS .... 16
14. NAP_RUNLIGHT ..... *NAP/RUNNING/LIGHT ON WORK
15. NAP_STATSLIB ..... *NAPSYSTEM/NSL ON WORK
16. NAP_LOGFAMILY ..... DISK

Action ▶+ ◀
Quit / Spcfy on entry or Enter Number / + (More) / Prev or Spcfy in
```

Enter 12 to change the SUP_MAXWHENS value.

```
                                           Metalogic S.a R.L.
                                           Install Utility
                                           54.540.18

                                           Supervisor variable - SUP_MAXWHENS

This variable specifies the maximum number of WHEN slots used
by Supervisor.
Supervisor must be restarted for this to take effect. If the
value is reduced to less than the current number of active WHENs
then some WHENs will be terminated.

Currently:un-defined

▶60◀

Its Default Value is :- 40 (Range 40 to 120)

Enter a value, Xmt to use the Default or SPCFY to return
```

Enter the value you want to use, in the example above 60, and transmit.

Now quit Install. The new limit will take effect the next time Supervisor restarts.

Take care if reducing this number as,if there are more slots in use than the limit, some could be lost on restart.

SUPERVISOR will display a waiting entry whenever an attempted WHEN or DO command causes the maximum number of WHEN slots to be exceeded. The waiting entry shown below will persist until manually acknowledged or SUPERVISOR is restarted.

ACCEPT:WARNING!!No slots available, WHEN Limit[40] has been exceeded!!

The waiting tasks is called:

(SUPERVISOR) SUPERVISOR/WHENLIMIT/EXCEEDED

DEFINE Command

The DEFINE command is used to maintain the library of OPAL programs within SUPERVISOR's *SCHEDULE file, to compile new programs and to query programs stored in the program directories that SUPERVISOR maintains. All OPAL programs of a particular type must be given a unique name (identifier) when they are DEFINEd. Each name (identifier) consists of 17 characters or less in length and may contain the underscore "_" character.

The DEFINE command creates, modifies, deletes and interrogates OPAL programs within the program dictionaries maintained by SUPERVISOR.

The syntax of the DEFINE command for query purposes is

DEF ? <OPAL program type> <program id>

To list all programs in the SITUation, ODTSequence, DISPlay and MEMO dictionaries enter:

TT DEF ? DEF

Each OPAL program is listed in alphabetical order within type, beginning with SITUations and continuing with ODTSequences, DISPlays and MEMOs.

The following are pages from a typical response:

```
TT +? DEF
-- SITUATIONS FOR * ----- Opal --- Last Used ----- Created ---- Ver
AC_CONTROL          LOGBOJ          520.93      Never          16:10,09/11/07
AC_CONTROL1         LOGEOJ          510.12      15:34,17/05/06 09:17,27/04/05
AC_LOG              LOG=15,3,2,1    510.25      12:11,18/05/06 09:36,13/10/05
AC_LOGCLOSE         FILECLOSE       510.12      12:17,18/05/06 09:17,27/04/05
AC_LOGON            LOGON           510.12      14:51,31/05/06 09:17,27/04/05
AC_LOGSECURITY      SECURITY        510.12      14:52,31/05/06 09:17,27/04/05
AC_QDSED            LOG=1,7         520.12      14:51,31/05/06 13:22,03/05/06
AC_TRICK2           PER=PK          510.12      10:16,04/05/06 09:17,27/04/05
AC_WHENOPT          MSG            510.12      14:52,31/05/06 09:17,27/04/05
AC_WHENOPT1         MSG            510.23      14:52,31/05/06 09:14,04/10/05
AH_COURSE_EX_4      MX=A           510.12      Never          09:17,27/04/05
AH_COURSE_EX_5      MX=W           510.12      Never          09:17,27/04/05
AH_COURSE_EX_61     MX=W           510.12      Never          09:17,27/04/05
AH_COURSE_EX_63     C              510.12      Never          09:17,27/04/05
AH_COURSE_EX_64     MSG            510.12      Never          09:17,27/04/05
AH_COURSE_EX_7      MX=L IBS,S,W,BOT.. 510.12      Never          09:17,27/04/05
AH_TEST            MX=A           510.12      Never          09:17,27/04/05
AL_TCPIPSEC         LOG=27,11      510.12      Never          09:17,27/04/05
ALAN_MAKEUSER       C              510.12      Never          09:17,27/04/05
AUTO_PG            PER=MT          510.12      Never          09:17,27/04/05
AUTOSN1            PER=MT          510.12      Never          09:17,27/04/05
Window S/I at DELL MCP
```


After transmitting TT + a few times:

TT +? DEF					
-- SITUATIONS FOR * -----					
		Opal	Last Used	Created	Ver
VERDB	DATABASE=OPEN	510.12	Never	09:19,27/04/05	
VL	VL	520.87	12:40,07/11/07	10:51,12/10/07	
VL_FIX	VL	520.16	Never	11:20,23/06/06	
VLD	VL	510.27	16:32,05/12/05	16:18,05/12/05	
ULTAPE	TAPEDB	520.16	Never	11:20,23/06/06	
ULTAPE2	TAPEDB	510.12	Never	09:19,27/04/05	
VLX	VL	520.16	Never	11:20,23/06/06	
W_DMUTILITY	MSG	510.12	Never	09:19,27/04/05	
WAITING_ENTRY	MX=W	510.12	Never	09:19,27/04/05	
WFLHELP	MX=W	510.12	Never	09:20,27/04/05	
X	LOG=29	520.56	12:20,14/02/07	12:17,14/02/07	
XREF	MX=A	999.14	09:12,20/03/08	15:51,21/10/02	
Y	MX=W	520.32	Never	10:55,24/10/06	
-- ODTSEQUENCES FOR * -----					
		Opal	Last Used	Created	Ver
AAAAA	SYSTEM	510.12	Never	09:17,27/04/05	
AABC	MESSAGE	510.12	Never	09:17,27/04/05	
AAC_CASE1	MESSAGE	510.12	Never	09:17,27/04/05	
AAC_CASE2	MESSAGE	510.12	Never	09:17,27/04/05	
AAC_CONCAT1	SYSTEM	510.12	Never	09:17,27/04/05	
AAC_CONCAT2	SYSTEM	510.12	Never	09:17,27/04/05	
Window S/1 at DELL MCP					

After some more pages:

TT +? DEF					
XX	SYSTEM	520.73	14:39,04/07/07	14:39,04/07/07	
X1	SYSTEM	520.56	12:17,14/02/07	12:05,14/02/07	
X2	SYSTEM	520.56	12:55,14/02/07	12:55,14/02/07	
Y	MIX	520.32	c11:04,22/01/08	10:55,24/10/06	
YY	MESSAGE	510.12	Never	09:20,27/04/05	
Z	SYSTEM	520.82	16:55,04/09/07	16:55,04/09/07	
ZAF_RESTART	SYSTEM	510.12	Never	09:20,27/04/05	
ZAL_LASTWORKDAY	TAPELABEL	520.65	Never	11:37,22/05/07	
ZAVAIL_RESTART	SYSTEM	510.12	Never	09:20,27/04/05	
ZBTS_RESTART	SYSTEM	510.12	Never	09:20,27/04/05	
ZX	MESSAGE	510.12	Never	09:20,27/04/05	
ZZ	SYSTEM	510.12	Never	09:20,27/04/05	
ZZT	SYSTEM	510.12	Never	09:20,27/04/05	
-- DISPLAYS FOR * -----					
		Opal	Last Used	Created	Ver
AC_HASH1	SYSTEM	510.12	16:43,15/05/06	09:17,27/04/05	
AC_HASH2	MESSAGE	510.12	16:45,15/05/06	09:17,27/04/05	
AC_HASH3	PER	510.25	16:52,15/05/06	11:25,07/10/05	
AC_VIA1	SYSTEM	510.12	Never	09:17,27/04/05	
ARC	MESSAGE	520.65	Never	11:37,22/05/07	
ARC2	MESSAGE	510.12	Never	09:17,27/04/05	
ARC3	SYSTEM	510.12	Never	09:17,27/04/05	
Window S/1 at DELL MCP					

And after some more we get to Memos:

TT +? DEF					
TESTIPP	SYSTEM	510.12	Never	09:19,27/04/05	
TPDB_BADDATE	TAPEDB	510.12	Never	09:19,27/04/05	
TPDB_CARTLABEL	TAPEDB	510.12	Never	09:19,27/04/05	
TPDB_CDSTATS	TAPEDB	510.15	Never	11:00,03/06/05	
TPDB_LABEL	TAPEDB	510.12	Never	09:19,27/04/05	
TPDB_REELLABEL	TAPEDB	510.12	Never	09:19,27/04/05	
TPDB_RF	PER	510.12	Never	09:19,27/04/05	
TPDB_STATS	TAPEDB	510.12	Never	09:19,27/04/05	
TPDB_SUMMARY	TAPEDB	510.12	Never	09:19,27/04/05	
TRACK_DISPLAY	SYSTEM	510.12	Never	09:19,27/04/05	
TRACY	SYSTEM	520.14	Never	09:09,16/05/06	
USE	SYSTEM	510.12	Never	09:19,27/04/05	
USERFN_1	SYSTEM	510.12	Never	09:19,27/04/05	
USERFN_2	SYSTEM	510.12	Never	09:19,27/04/05	
X	SYSTEM	520.95	Never	12:08,29/11/07	
XX	WINDOWS	500.62	14:19,18/04/07	13:44,18/04/07	
X1	SYSTEM	510.12	Never	09:20,27/04/05	
Y	MIX	520.32	Never	10:55,24/10/06	
-- MEMOS FOR *		Opal	Last Used	Created	Ver
AC_CONCAT		510.21	Never	16:14,25/08/05	
AC_HASH		510.21	Never	16:18,25/08/05	

And then COMMands

TT +? DEF					
AC_HASH		510.21	Never	16:18,25/08/05	
AC_KEYIN		510.21	Never	16:21,25/08/05	
AC_LIST		510.12	Never	09:17,27/04/05	
AC_OBJECTS		510.12	Never	09:17,27/04/05	
AC_SHOW		510.12	Never	09:17,27/04/05	
AC_SPLIT		510.12	Never	09:17,27/04/05	
AC_TRICK		510.12	Never	09:17,27/04/05	
AC_VAR		510.12	Never	09:17,27/04/05	
DB_NOTES		510.12	Never	09:18,27/04/05	
META_SUPPORT		510.12	Never	09:18,27/04/05	
REC_W		510.15	Never	16:55,24/05/05	
TEST_INVOKE		520.99	Never	10:08,22/01/08	
TPDB_CONTEXTS		510.12	Never	09:19,27/04/05	
X		510.12	Never	09:20,27/04/05	
-- COMMANDS FOR *		Opal	Last Used	Created	Ver
C	MESSAGE	500.62	09:54,17/03/08	14:51,27/04/07	
MAKE_LIVE	MESSAGE	520.34	16:53,05/03/08	17:00,26/10/06	
NET	SYSTEM	520.07	15:10,16/02/06	15:10,16/02/06	
NETWORK	SYSTEM	520.07	10:02,16/02/06	13:47,14/02/06	
NEW_DBS	MESSAGE	510.27	Never	11:05,08/11/05	
Y	MIX	520.32	08:18,13/03/08	10:55,24/10/06	

Window S/1 at DELL MCP

Finally:

DEF ? DEF					
Y	MIX	520.32	08:18,13/03/08	10:55,24/10/06	
-- ODTSEQUENCES for		Opal	Last Used	Created	Ver
BOB	MESSAGE	520.12	11:18,21/04/06	11:18,21/04/06	
SWITCH	MESSAGE	500.37	Never	15:17,09/02/05	
TEST	SYSTEM	510.15	Never	09:43,25/05/05	
-- ODTSEQUENCES for		Opal	Last Used	Created	Ver
AAAAA	SYSTEM	510.12	Never	09:17,27/04/05	
ADHOC_MENU	MESSAGE	510.12	Never	09:17,27/04/05	
X	MESSAGE	510.12	Never	09:17,27/04/05	
-- ODTSEQUENCES for		Opal	Last Used	Created	Ver
TEST	MESSAGE	510.12	Never	09:17,27/04/05	
-- ODTSEQUENCES for		Opal	Last Used	Created	Ver
X	SYSTEM	510.12	Never	09:17,27/04/05	
-- ODTSEQUENCES for		Opal	Last Used	Created	Ver
ARC	MESSAGE	510.12	Never	09:17,27/04/05	

Window S/1 at DELL MCP

The following response fragment from SUPERVISOR will list all programs contained within the SITUation Dictionary:

```
TT +? SITU
-- SITUATIONS FOR * ----- Opal --- Last Used ----- Created ---- Ver
AC_CONTROL          LOGBOJ          520.93      Never        16:10,09/11/07
AC_CONTROL1         LOGEOJ          510.12      15:34,17/05/06 09:17,27/04/05
AC_LOG              LOG=15,3,2,1    510.25      12:11,18/05/06 09:36,13/10/05
AC_LOGCLOSE         FILECLOSE       510.12      12:17,18/05/06 09:17,27/04/05
AC_LOGON            LOGON           510.12      14:51,31/05/06 09:17,27/04/05
AC_LOGSECURITY      SECURITY        510.12      14:52,31/05/06 09:17,27/04/05
AC_QDSED            LOG=1,7         520.12      14:51,31/05/06 13:22,03/05/06
AC_TRICK2           PER=PK          510.12      10:16,04/05/06 09:17,27/04/05
AC_WHENOPT          MSG             510.12      14:52,31/05/06 09:17,27/04/05
AC_WHENOPT1         MSG             510.23      14:52,31/05/06 09:14,04/10/05
AH_COURSE_EX_4      MX=A            510.12      Never        09:17,27/04/05
AH_COURSE_EX_5      MX=W            510.12      Never        09:17,27/04/05
AH_COURSE_EX_61     MX=W            510.12      Never        09:17,27/04/05
AH_COURSE_EX_63     C               510.12      Never        09:17,27/04/05
AH_COURSE_EX_64     MSG             510.12      Never        09:17,27/04/05
AH_COURSE_EX_7      MX=L IBS,S,W,BOT.. 510.12      Never        09:17,27/04/05
AH_TEST            MX=A            510.12      Never        09:17,27/04/05
AL_TCPIPSEC         LOG=27,11       510.12      Never        09:17,27/04/05
ALAN_MAKEUSER       C               510.12      Never        09:17,27/04/05
AUTO_PG            PER=MT          510.12      Never        09:17,27/04/05
AUTOSN1            PER=MT          510.12      Never        09:17,27/04/05
Window S/1 at DELL MCP
```

There may be a need to limit the programs listed. A partial list can be obtained by using a wildcard pattern in the program name field of the command. The equal sign (=) means match any string where it occurs.

The DEFINE interrogation form of this command:

TT DEF ? DEF =EXT=

produces a listing of all programs in all dictionaries containing the text "EXT".

```
DEF ? DEF =EXT=
-- SITUATIONS FOR * ----- Opal --- Last Used ----- Created ---- Ver
ICEXTRACT_EOJ       LOGEOJ=E0J      510.12      Never        09:18,27/04/05
LOG_EXT1            LOG=1,7         510.12      Never        09:18,27/04/05
LOG_EXT2            LOG=2,12        510.12      Never        09:18,27/04/05
LOG_EXT33           LOG=33,2        510.12      Never        09:18,27/04/05

-- ODTSEQUENCES FOR * ----- Opal --- Last Used ----- Created ---- Ver
HOTEXTINT           MESSAGE         510.12      Never        09:18,27/04/05
HTTP_CONTEXTLIST    SYSTEM         520.42      c12:17,24/11/06 12:17,24/11/06
ICEXTRACT_CHG       LOGEOJ          510.12      Never        09:18,27/04/05
LOG_EXT1            LOG             510.12      Never        09:18,27/04/05
LOG_EXT2            LOG             510.12      Never        09:18,27/04/05
LOG_EXT33           LOG             510.12      Never        09:18,27/04/05
START_IVR_EXTRACT   SYSTEM         510.12      Never        09:19,27/04/05
TH_EXT             MESSAGE         520.14      Never        14:43,17/05/06

-- DISPLAYS FOR * ----- Opal --- Last Used ----- Created ---- Ver
C_DATETOTEXT        MESSAGE         510.12      Never        09:17,27/04/05
META_NEXTSCRATCH    SYSTEM         520.73      Never        11:35,29/06/07

-- MEMOS FOR * ----- Opal --- Last Used ----- Created ---- Ver
TPDB_CONTEXTS       510.12         Never        09:19,27/04/05
Window S/1 at DELL MCP
```

Listing or checking for the presence of an OPAL program is quite simple:

TT DEF ? <type of OPAL program> <program id>

If the program does not exist within the type specified, SUPERVISOR responds:

NAME OF A <type of OPAL program> EXPECTED

If SUPERVISOR finds a program with that id then the program will be displayed.

If the following is entered:

TT DEF ? SITU NOFILE

then SUPERVISOR might respond with:

```
TT DEFINE + ODTSEQUENCE NOFILE(MX):  
  ODT("COPY *",DECAT(DECAT(RSVP,"NO FILE ",1)," ON DEV",4),  
    " FROM DEV(PACK,HOSTNAME=POWEREDGE)");
```

It is important to note that SUPERVISOR has replaced the question mark (?) in the command with a plus sign (+) in the response to make it easier to modify the program. Care must be taken since this plus sign (+) means that no check is made to prevent overwriting of an existing program. If Supervisor option 30, DEFINERESP, is set, the question mark is not substituted and must be changed to update the program.

Also, when using the abbreviated form of the command and/or the type of OPAL program, SUPERVISOR will change the abbreviations to their full spelling. DEF was changed to DEFINE, SITU changed to SITUATION.

To delete an OPAL program or a MEMO (see next section) from a directory, the "DEF -" syntax is used. SUPERVISOR will respond with the message

```
<type of OPAL program> DEFINITION DELETED
```

A typical request for a program to be deleted might be:

```
TT DEF - SITU III_MIX_FINDER_X
```

And Supervisor would respond with

```
SITUATION DEFINITION DELETED
```

Sample OPAL Program

When creating a new OPAL program, it is good practice to first check for the presence of the same type of OPAL program with the same program id. A simple example of creating a new OPAL program follows. (Do not be concerned about understanding this program; it will be covered later in this section.) Programs can be entered in free form beginning at the home position and transmitting after the last statement.

This particular OPAL program is of type SITUation. (Remember, a SITUation describes a condition to look for.) The name of this OPAL is MIX_FINDER and it will look for all the programs in the mix with "METALOGIC" somewhere in the title. At the completion of this program we will have a list of mix numbers of tasks where this condition has been met.

If you wish to enter this program in your system, you might want to prefix the program name with your initials (ABC_ in the text below) so as to avoid overwriting any existing programs.

```
TT DEF SITU ABC_MIX_FINDER_X (MX) :  
  "METALOGIC" ISIN NAME
```

SUPERVISOR will respond with:

```
TT DEF SITU ABC_MIX_FINDER_X (MX) :  
  "METALOGIC" ISIN NAME  
SITUATION DEFINITION ENTERED
```

Note in this example we did not use a plus sign (+). This is done for safety. Since we are attempting to create a new OPAL program, we don't want to accidentally overwrite an existing one.

Had there been a SITUation named ABC_MIX_FINDER_X, the response would have been

```
TT DEF SITU ABC_MIX_FINDER_X (MX) :  
  "METALOGIC" ISIN NAME  
That SITUATION name has been DEFINED already
```

telling us to pick a different name.

Never put in the plus sign (+) yourself – at least not at first. Use the inquiry command

```
DEF ? <type of OPAL program> <program id>
```

and let SUPERVISOR put in the plus sign (+) for you. It's safer!

To modify an existing program, first do an inquiry of the program using the "TT DEF ?" syntax, e.g.

```
TT DEF ? SITU ABC_MIX_FINDER_X
```

SUPERVISOR will respond by listing the program and substituting a plus sign (+) in place of the question mark (?). Modify the program and then transmit from the end of the program. Remember to place the cursor anywhere after the last statement. SUPERVISOR will respond with

```
<type of OPAL program> DEFINITION ENTERED
```

OPAL programs are input as a single screen transmission or from a CANDE file.

When creating or modifying an OPAL program, the choice of indentation style is yours. OPAL programs are always free form. You can edit up to 24 lines directly on the ODT or REMOTESPO screen. When the 24-line limit is reached or if the program becomes too cumbersome, you can save the last compiled version to a disk file, and edit it with CANDE or EDITOR. If you expect a SITUation, ODTSequence or DISPlay program to exceed the 24-line screen limit, you can use a CANDE file from the beginning. The CANDE file should begin with the DEFINE command, e.g.

```
TT DEF SITU ABC_MIX_FINDER_X (MX) :
```

and end with a back slash "\". Note that the use of the TT' prefix is optional here since it is treated as noise when the command is processed by SUPERVISOR.

The full syntax for the Define can be found [here](#)

See [ENTER](#)

Note that the SUPERVISOR COMS windows implementation does in fact support larger screen dimensions up to a maximum of 63 lines per page and 132 characters per line.

The OPAL compiler expects input in record-oriented format (even if the source of a program is a full screen transmission) and each logical 'line' is 80 characters in length.

From a SUPERVISOR window with screen widths longer than 80 characters any DEFINE that attempts to use the area beyond character 80 will give an error:

Text overflow at line 6

Comments in OPAL programs are introduced by the characters `/*` (forward slash, asterisk) and terminated by the characters `*/` (asterisk, forward slash). The more commonly used percent (`%`) character may also be used to introduce a comment on a single line. The ENTER command is used to load OPAL programs that have been externally created into the `*SCHEDULE` file.

```
TT DEF SITU ABC_MIX_FINDER_X (MX) :  
  % This is a comment line  
  "METALOGIC" ISIN NAME
```

To delete an OPAL program or a MEMO from a directory, the "DEF -" syntax is used. SUPERVISOR will respond with the message

```
<type of OPAL program> DEFINITION DELETED
```

If the program was created from a CANDE file, only the entry in the directory of OPAL programs are removed in SUPERVISOR while the CANDE file remains intact. A typical request for a program to be deleted would appear as:

```
TT DEF - SITU ABC_MIX_FINDER_X
```

SUPERVISOR's response would be

```
SITUATION DEFINITION DELETED
```

MEMOs

A MEMO is documentation which is created and modified by the DEFINE command and is primarily used for operator notification of new or altered procedures, a site specified HELP facility, and messages. When creating or modifying a MEMO, the choice of indentation style is yours. MEMOs are always in a free form format.

For example

```
TT DEF MEMO META_SUPPORT:  
  STIRLING: +44 1786 445014  
  LUXEMBOURG: +352 326850
```

In addition to the **DEF ? MEMO** syntax for inquiries of MEMOs, entering:

```
TT MEMO ?
```

will display a list of all the identifiers of all the current MEMOs at the requesting terminal.

The above MEMO is viewed by entering:

```
TT MEMO META_SUPPORT
```

EVALUATE Command

The EValuate command is used to monitor, execute, and stop SITUations. The command:

```
TT EV ?
```

produces a display of the current status of all running OPALs. It is basically equivalent to the ODT command MX for normal MCP tasks.

The response should look something like this:

```
TT +
----- SUPERVISOR WHEN STATUS (Limit = 40, Active = 21, Queued = 0) -----
W 000 04759 WHEN BOB_CLEANUP DO BOB_CLEANUP RUNNING 01
    FOR BOB TIMES:CPU=00:00:00,IO=00:00:00,ET=02:09:03
    34 evals, 1 ODTs entry, 341 Restarts
R 000 04758 DO REC_STATUS (WAIT TO 11:23.00) 02
    TIMES:CPU=00:00:04,IO=00:00:00,ET=02:09:03
    0 evals, 1 ODTs entry, 92 Restarts
W 000 04757 WHEN META_SLOTCHECK DO META_SLOTCHECK RUNNING 03
    TIMES:CPU=00:00:00,IO=00:00:00,ET=02:09:04
    11 evals, 38 Restarts
018 04756 WHEN META_MSGMON DO META_MSGMON RUNNING 04
    TIMES:CPU=00:00:00,IO=00:00:00,ET=02:09:04
    155 evals, 106 Restarts
002 04755 WHEN BOB_WEBSECURITY DO BOB_WEBSECURITY RUNNING 05
    TIMES:CPU=00:00:00,IO=00:00:00,ET=02:09:04
    68 evals, 206 Restarts
019 04754 WHEN REC_C DO REC_C RUNNING 06
    TIMES:CPU=00:00:02,IO=00:00:04,ET=02:09:04
    94 evals, 94 ODTs entries, 203 Restarts
018 04753 WHEN META_MAKELIVE DO META_MAKELIVE RUNNING 07
    TIMES:CPU=00:00:00,IO=00:00:00,ET=02:09:04
    155 evals, 106 Restarts
Window S/I at DELL MCP
```

Each active Opal program runs in a 'When Slot'. In the screen above there are 21 active slots with a limit on the total slots of 40. The last column indicated the slot number for each active Opal.

To execute a SITUation, the EV command will be in one of these forms:

```
TT EV <SITUation def>
TT EV <SITUation def> DISPlay <DISPlay def>
TT EV <SITUation def> DO <ODTSequence def>
TT EV <SITUation def> TEST <ODTSequence def>
```

A <SITUation def>, <DISPlay def>, and a <ODTSequence def> are either the names of OPAL programs already stored in SUPERVISOR's SCHEDULE file or the actual program statements themselves enclosed within parentheses. The latter form is referred to as an In-Line Opal.

DO Command

The DO command is used to run or stop an ODTSequence without a controlling SITUation. The ODTSequence must be DEFINEd as type MX, PER, MSG, SYSTEM, PRINTS, TAPEDB or WHEN. Parameters may be required to pass information to the ODTSequence.

Context	Parameters
MX	Mix number list
PD	File or Directory specification
PER	Unit number list
MSG	Any text
SYSTEM	None required
TAPEDB	Serial number list
PRINTS	Request number list
WHEN	Slot number list

TT DO <ODTSequence def>

is the syntax used to execute the ODTSequence unconditionally.

The <ODTSequence def> is either the name of an ODTSequence already stored in SUPERVISOR's SCHEDULE file or the actual program statements themselves enclosed within parentheses. This form is known as an In-Line Opal.

TT DO + <ODTSequence def>

will first check for a running DO of the same name. If one is running SUPERVISOR will respond with

ODTSEQUENCE ALREADY RUNNING

Otherwise the ODTSequence will begin to execute.

The "DO –" (minus sign) form of the DO command is used to terminate an ODTSequence that was started with a DO or TEST. SUPERVISOR will respond with the message:

THE DO WILL BE DEACTIVATED

TEST Command

The TEST form of the DO command is used to debug an ODTSequence. In TEST mode, system commands are not passed on to CONTROLLER but are displayed as messages on the ODT. For example:

TT TEST HLSTUFF

will cause the ODTSequence HLSTUFF to begin executing. SUPERVISOR will display

THE DO WILL BE DONE

at the beginning of the program's execution. All system commands will then be displayed.

To view all the displays, use the system's MSG command:

```
---Mix-Time----- MESSAGES -----
* 7302 13:43 SUPERVISOR/HLSTUFF ??RUN:SYSTEM/CANDE
* 7302 13:43 SUPERVISOR/HLSTUFF ODT:ID 112
* 7302 13:43 SUPERVISOR/HLSTUFF ??RUN:METALOGIC/FLEX/LIBRARY
* 7302 13:43 SUPERVISOR/HLSTUFF ??RUN:METALOGIC/DBCONTROL
* 7302 13:43 SUPERVISOR/HLSTUFF ODT:TT SAVE SCHEDULE
```

DISPlay (/) Command

The DISPlay or / (single forward slash) command executes a DEFINEd DISPlay and sends the result to the terminal. It is a way to call a new user-specified ODT information message. The DISPlay must be DEFINEd as type MX, PER, MSG, SYSTEM or TAPEDB, or PRINTS. Parameters to DISPlays are identical to those for ODTSequences in the DO command above.

For example, MIX_RESULTS is the name of a DISPlay of type MX. When the DISPlay context is MX, a mix number list is required.

If this DISPlay were to be invoked and no mix number list was provided SUPERVISOR would respond with an error message:

MORE INPUT REQUIRED

Assuming that mix numbers 7725, 7745, 7706, 7712 are valid, then if the following command is submitted to SUPERVISOR:

```
TT DISP MIX_RESULTS 7725,7745,7706,7712
```

the following information might be displayed:

```
TT / MIX_RESULTS 7725,7745,7706,7712
```

7725	*SYSTEM/COMS	00:00:09	00:12:59	1%	4040	6232
7745	(SUPERVISOR) SUPERVISOR/REMOTESPO	00:00:00	00:05:03	0%	6011	6072
7706	(SUPERVISOR) METALOGIC/SUPERVISOR/EV	00:00:01	00:14:23	0%	5580	5704
7712	*METALOGIC/DBCONTROL	00:00:01	00:14:01	0%	2712	3700

WHEN Command

The WHEN command controls the linkage of OPAL SITUations and ODTSequences or DISPlay programs. The ONCE command is a variant of the WHEN command in that whilst WHEN continues running until it is deactivated, ONCE stops executing after the first time the SITUation evaluates to TRUE. There are two programs specified in the WHEN command:

- SITUation and DISPlay
- SITUation and ODTSequence.

SUPERVISOR responds to a WHEN command with a message like:

```
SUPERVISOR <program id>+<program id>:BEGIN MONITORING
```

An active WHEN is deactivated by breaking the linkage between the SITUation and ODTSequence or DISPlay. Either the EV or WHEN command may be used:

```
EVAL MY_SITU DO  
WHEN ANOTHER_SITU DO  
WHEN MY_SITU DISP
```

SUPERVISOR will respond immediately with a message:

```
THE WHEN WILL BE DEACTIVATED
```

When the program terminates, SUPERVISOR will respond with a termination message:

```
TT EV HI_IOT DO  
SUPERVISOR/HI_IOT+ HI_IOT [6876]:NORMAL TERMINATION
```

ENTER Command

► **<file name>** IN **<file name>** ON **<family>**

The ENTER command is similar to the CANDE DO command. The file must contain valid SUPERVISOR TT commands. Each command can be input in a free form format and must be separated from the next command with a back slash (\). The 'SAVE AS' command will produce a file in this format.

The modifier after ENTER may be used to filter what is entered.

The IN syntax is used to enter from a symbol file which is in a container.

Ex

```
ENTER FROM OPALS/TEST IN "OPALS.CON" IN "META.CON" ON MYFAM  
ENTER REF OPALS/TEST IN "REF.CON"  
ENTER ODTs BOB_TEST FROM OPALS/BOB IN "BOB.CON"
```

Schedule or no modifier

Will enter all commands from the specified file

Trusted

Will enter all commands from the specified file but any WFL command in AFTERs will not be checked. This allows the saved schedule from another system to be added even if some families specified in After commands are not yet available.

AFTER

Only AFTER commands will be processed

WHEN

Only commands which would run something in a WHEN slot are processed.

DEF

Only DEFINE commands are processed.

ODTS

Only DEFINE ODTS commands are processed.

SITU

Only DEFINE SITU commands are processed.

DISP

Only DEFINE DISP commands are processed.

MEMO

Only DEFINE MEMO commands are processed.

<pattern>

For modifiers where a pattern list is allowed then only defines which match a pattern in the list are entered. If a pattern is supplied by no program type then DEF is assumed.

To load the example OPALs for this chapter, enter the following command:

```
TT ENTER DEF = FROM OPALS/SUPERVISOR/GETTINGSTARTED
```

SUPERVISOR will immediately respond with

```
SUPERVISOR: INITIATING ENTER FROM OPALS/SUPERVISOR/GETTINGSTARTED
```

Other Examples

```
ENTER ODTS TEST_1,TEST_2 FROM *OPALS/TEST IN "TEST.CON" ON DEV  
ENTER WHEN FROM (LIVE)SYMBOL/SCHEDULE ON MYFAM  
ENTER DEF BOB_ = FROM (LIVE)SYMBOL/SCHEDULE ON MYFAM
```

When any ENTER process completes, SUPERVISOR sends a report to your terminal detailing which OPAL programs (if any) failed and why, plus a summary line

indicating the totals of success and failure.

The above command should generate a response like this:

```
SCHEDULE AMENDED,795 RECORDS PROCESSED,62 COMMANDS HANDLED
62 DEFINES FOUND, 62 OPALS COMPILED OK
ENTER COMPLETED
```

Each ENTER produces its own backupfile listing which shows all compiled OPALs, reporting all syntax errors where appropriate.

The following shows the first few lines of the GETTINGSTARTED file:

```
NEXT+      ....*....1....*....2....*....3....*....4....*....5....*....
6....*....7....*....8
00000100    DEFINE + SITUATION AUTO_PG(PER=MT) :
00000200    % GETTING STARTED: Auto purging of tapes
00000300 NOT SCRATCH AND TLPGOK(SERIALNO) AND
00000400 WRITEENABLED AND
00000500 TLTITLE(SERIALNO) =FILEID(LABEL,1)
00000600\
00000700    DEFINE + ODTSEQUENCE AUTO_PG(PER) :
00000800    % GETTING STARTED: Auto purging of tapes
00000900    ODT("PG MT ",UNITNO);
00001000    ODT("TT TP ",SERIALNO," NOTE");
00001100\
```

OPAL Example

This section will walk through an example of an OPAL program. On some sites with easy access to privileged terminals, users are inclined to increase their task's priority when they become impatient. This program will restrict this behaviour. As before, you might wish to use your own initials to replace "iii".

On most sites, user programs come in with a PRIORITY of 50. This gives all these programs a fairly even shot at the processor. If someone puts up their priority, the MCP will always give the processor to that task when it asks for it, over one of lower priority. In other words, raising a task's priority is normally equivalent to reducing the access to the CPU of all tasks with a lower priority – somewhat anti-social behaviour. When a task's priority is changed, the OPAL MX=PR event is caused.

The following is a definition of a SITUation, which checks for task priority changes:

```
TT DEF SITU METAPRI (MX=PR) :
    PRIORITY > 50
```

In the above example, TT is the SUPERVISOR command prefix, DEF is the abbreviation for the DEFINE command. (Remember that if we had used a "+" (plus sign) between DEF and SITU, that would allow the overwriting of an existing OPAL program with the same name.) SITU is the type of OPAL program and could also have been entered as SITUATION. METAPRI is the OPAL Identifier of this SITUation. The modifier(MX=PR) is the context or class in which the system information accessed is based. The DEFINE "header" then ends with a colon.

Our program contains one simple expression: `PRIORITY > 50`.

This will cause the `SITUation` always to accept the event whenever the priority change is greater than 50. "PRIORITY" is an attribute of the object class `MX`, which returns the priority after the change. ">" means "greater than" and "50" is an arithmetic constant. Remember "TT HELP ATT PRIORITY" is used to access information about the `ATtribute` `PRIORITY`.

To test the `OPAL SITUation` program just written, use the `Evaluate` command. Enter

```
TT EV METAPRI
```

`SUPERVISOR` will respond immediately with the message:

```
THE EVALUATION WILL BE DONE
```

When completed, `SUPERVISOR` will display the following results (the mix number will of course be different on your system)

```
TT EV METAPRI
```

```
SUPERVISOR/METAPRI:RETURNED TRUE FOR 7 MIXNUMBERS:7750,...
```

Next, create an `OPAL ODTSequence` program. An `ODTSequence` is a sequence of statements that describe an action to be taken.

```
TT DEF ODTs METAPRI (MX) :  
    ODT (MIXNO,"PR",100-PRIORITY) ;  
    DISPLAY("PRIORITY SET TOO HIGH") ;
```

Note the duplication of the program identifier of `METAPRI`. Since each type of `OPAL` program is kept in a separate and distinct dictionary, duplication of names is allowed. A `SITUation` and `ODTSequence` to be run together are often named the same for convenience, although this is not required.

The context of this new `ODTSequence` is `MX` whereas in the `SITUation` above we limited ourselves to looking only at the priority changes by defining the context as `MX=PR`. The program contains two statements.

The first sends a `PR` command to the `MCP`, specifying that the priority should be "complemented". Thus if the new priority is 99, it will be set to 1, 80 to 20, 51 to 49. Effectively, the punishment is more severe the higher the priority was set to. The second statement makes a `DISPLAY` informing the operator and user. `DISPLAYs` actioned in `MX ODTSequences` act as if the program itself had done the `DISPLAY`, so that it comes out in the Job Summary, on the terminal, etc.

To test the `ODTSequence` in the above example, we will use the `TEST` command. Enter

```
TT TEST METAPRI <mix number>
```

where `<mix number>` is the mix number of a task with priority higher than 50 on your system. `SUPERVISOR` will respond immediately with the message

```
THE DO WILL BE DONE
```

Each ODT Statement will then be displayed. The results will appear something like the example below with the last ODT Statement displayed:

```
TT TEST METAPRI 7750
7750 SUPERVISOR/METAPRI ODT:7750PR20
```

A test before a live execution is especially important when you can affect the entire system. In the above examples, we tested the SITUation with the EVALuate command and the ODTSequence with the TEST command. To test both programs together enter

```
TT EV METAPRI TEST METAPRI
```

EV will EVALuate the SITUation once and pass the results to the ODTSequence. TEST will display the KEYIN strings in the ODT Statements instead of actually executing them. If any statements cause an error, an appropriate message will be displayed. Again, use the MSG system command to see the results.

To finally put this "rule" into action, we would enter:

```
TT WHEN METAPRI DO METAPRI
```

This will now appear in the EV ? report.

In the real world, there are some users, who are too important, powerful, or maybe even vexatious, to restrict in this way. So we can modify the SITUation METAPRI by making an exception of, for example, the users "UNRUH" and "CLINTON".

To modify the SITUation, we enter:

```
TT DEF ? SITU METAPRI
```

SUPERVISOR will display the SITUation.

Modifications can now be made. If we want to exempt users with the usercodes of UNRUH and CLINTON then the third line of the example below – note the "curly" braces { and } – should be added to the program.

And we might also want to exempt MCSes.

```
TT DEF SITU METAPRI (MX=PR) :
    PRIORITY > 50 AND NOT MCS AND
    NOT USER = {"UNRUH", "CLINTON"}
```

Transmit the entire program.

We will be warned that:

```
Code is being used in slot <slot number>
```

Our new code will not be used in the active slot until it has been stopped and restarted

We can still re-test

```
TT EV METAPRI TEST METAPRI
```

The WHEN can then be stopped and restarted to use the new code:


```
TT WHEN METAPRI DO
TT WHEN METAPRI DO METAPRI
```

To get more information on the USER or MCS ATtributes, do a TT HELP ATT USER or a TT HELP ATT MCS. The "USER = { ... }" syntax means USER is equal to any of the choices within the curly braces.

Basics

This section explains the basics – how to communicate with the software, where to expect responses, common command syntax, and so on. Most of the SUPERVISOR commands can take various modifiers, for example the FOR modifier which "attaches" a usercode to the command. These modifiers are explained in this section.

CTables Basicsommands (or modifiers) dealt with in this section are:

Commands	Synonym	With AF?
EH	–	No
FOR	–	No
HELP	TEACH	No
NS	–	No
PRINT	PR	Yes
TT	–	Yes
VIA	–	Yes

Railroad Diagrams

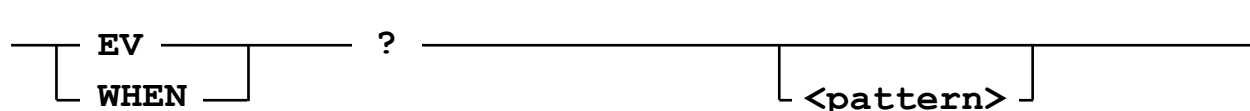
Railroad diagrams are used throughout this manual to describe the syntax of the various SUPERVISOR commands. Many of these can become quite complicated and confusing due to the number of options available within some commands.

In order to simplify the diagrams to a degree, a convention has been adopted for commands that have synonyms that are abbreviations of the full command keyword. For example, AF is a valid synonym for the AFTER command.

To show this, the letters comprising the valid contracted synonym are shown in upper case and the rest of the keyword shown in lower case:



Where a command has an alternative that is not an abbreviation, then the alternative is shown as a branch on the diagram:



In the diagram for the command above, there are two possible alternative keywords: EV, and WHEN.

Communicating with SUPERVISOR

From the ODT

In order for SUPERVISOR commands to be recognised when they are input at a system console, they must be prefixed by the "TT" system command. The TT command (in a non-SUPERVISOR system) is a simple "Text Test" function whereby any text after "TT" is written to the SUMLOG and the following message returned to the originating ODT:

TEST FUNCTION

SUPERVISOR "intercepts" these TT inputs by requesting notification from the MCP of log entries. When a log entry of type "TT" is found, SUPERVISOR retrieves the text and treats it as an input command.

Since the standard response of "TEST FUNCTION" (message number 180245 on some MCPs) from a TT command may be considered confusing, it is possible to use the standard Unisys MESSAGE TRANSLATION UTILITY to change it to something else (for example, the message could be changed to "Passed to SUPERVISOR"). Please refer to the **Unisys A Series Message Translation Utility User's Guide** for detailed explanations on how to accomplish this.

Although the TT prefix is required only when using an ODT as input, it is good practice to follow this convention even when entering commands from other sources. The standard adopted throughout this manual is: all commands intended for SUPERVISOR will be prefixed with TT, commands intended for system CONTROLLER will not.

An alternative and more primitive method of reaching SUPERVISOR from the ODT is to pass the command via an SM Message of the form <mixno> SM <command>. The response is returned, by default to the lowest numbered ODT, but that can be permanently changed with the USE ODT command. It may also be changed for an individual command by using the VIA modifier.

From MARC

METALOGIC provides a file on the standard release tape, which can be used to establish a directive in MARC. Commands prefixed by TT may be entered through a MARC menu and responses are returned by MARC using its standard screen format. The file is called:

METALOGIC/SUPERVISOR/TTINTERFACE

To establish this file as the codefile for the directive, the MARC DIRECTIVE command is entered as:

```
DIRECTIVE TT = *METALOGIC/SUPERVISOR/TTINTERFACE:SYSTEM
```

If this is successful, MARC will return a message:

The DIRECTIVE "TT" is established and available.

In order to be able to use the DIRECTIVE Command, it must be input on a terminal which is logged in under a SYSTEMUSER capable usercode or is already a SYSTEMUSER station.

The program runs as an MCS once it has been established as the MARC directive. Access rights to the TT Directive are specified when the MARC DIRECTIVE command is entered – refer to the **Unisys MARC Operations Guide** for details of the complete command syntax.

An ODTS running via the MARC directive is restricted in responses it can send via the SHOW statement.

Any output is returned when the ODTS waits for a total of more than 10 seconds or ends. If the ODTS waits for more than a total of 10 seconds then no further output will be sent.

This restriction has been relaxed slightly. If a wait is for time and the total time waited is less than 10 seconds then the output will not be sent until the end of the ODTS, any non timed wait or a timed wait which takes the total waited time above 10 seconds.

Ex. SHOW ("ONE") ; WAIT (5) ; SHOW ("TWO") ; WAIT (6) ; SHOW ("THREE")

In an ODTS started via TT Interface would return:

ONE

TWO

From a SUPERVISOR COMS WINDOW

Probably, the most convenient everyday method of communicating with SUPERVISOR is with a COMS Window. The typical Window declaration in the COMS Utility would be as follows:-

MAX_USERS	= 0
MAX_DIALOGS	= 1
MAX_TRANCODE_SIZE	= 0
MCS	= Y
TRUNCATED_RESULTS	= N
TITLE	= *METALOGIC/SUPERVISOR
NOTIFY_OPEN	= N
NOTIFY_ON	= N

If NOTIFY_OPEN or NOTIFY_ON are set to Y, appropriate Supervisor commands e.g. WS should be specified for the associated text.

The number of concurrent Windows allowed from SUPERVISOR, by default, is set to 15. This value can be modified by changing the configuration variable SUP_MAXWINDOWS. The maximum is 16 while the minimum is 0. It should be noted that the setting of this variable also influences the number of Supervisor REMOTESPOs made available. A total of 16 Windows and REMOTESPOs can

currently be supported, so if SUP_MAXWINDOWS is set to 10, then 6 REMOTESPO sessions will be available.

The setting can be changed using the Metalogic INSTALL utility run with the CONFIG parameter and choosing the Supervisor sub menu.

U META/INSTALL CONFIG

Another configuration variable, SUP_WINDOWSECURITY controls SUPERVISOR's handling of window security. It can currently take one of two values – MINIMUM or MAXIMUM (the default). MAXIMUM means that the Station must be logged on under a privileged or systemuser usercode. If the variable is set to MINIMUM, Supervisor will not perform any checks on the usercode and log-ons to Supervisor Windows will be controlled entirely by the standard COMS security mechanisms.

The USE command reports the Window Security level in use and allows this to be changed using the USE WINDOWSECURITY <level> command. Changing the level of Window Security will not affect current Window sessions but will be used to validate any new log-on attempts.

The number of available Window and Remotespo sessions are reported by the 'WINDOW ?' and 'REMOTESPO ?' commands.

Input from COMS windows-originated commands will accept a prefix of TT but it is not required. In fact some commands do 'collide' with MCP variants and to use the MCP command a prefix of ODT is required. Some MCP commands that require an ODT prefix are:

- DO
- RESTRICT
- PRINT
- SAVE
- SHOW
- TL
- WINDOWS

SUPERVISOR WINDOWS are intended to act as normal, unrestricted ODTs, so all system commands may be input (except ADM and ?? commands). SUPERVISOR provides its own TERM command for configuring the terminal in use. Although the standard ADM cannot be used on a WINDOW, a WHEN command linking a SITUation to a DISPlay provides much of the functionality of ADM, with much better filtering to ensure that only events important to the terminal operator are reported. SUPERVISOR will simulate a ??RUN command as an activity scheduled with the AFTER command.

However, the ?? system commands are really intended for low-level interaction between the system operator and the MCP in emergency situations, so it would not

be appropriate for SUPERVISOR to attempt to handle these as non-ODT input.

Following the normal convention a ??WRU command may be input on a Window Station and will return a response giving details of that Station.

It is possible to allow restricted access to Supervisor Windows.

First define an Installation Data item where String1 is set to the ODTs you will use to process input from the window. If the ODTs defined in the installation data String1 field is preceded by () then the ODTs will run under the usercode of the window. Without a () prefix the ODTs run with no usercode.

EX.

ID - INSTALLATION DATA ACTIVITY		COMS					
Action:	CR	MODIFY	INQUIRE	DELETE	GO	HOME	(Press SPCFY for help)
	SE	FI	LA	NE	PR		
Installation Data Name	SCHED						
Valid Window Name	ALL						
Integer1	0						
Integer2	0						
Integer3	0						
Integer4	0						
Hexadecimal1	NONE						
Hexadecimal2	NONE						
String1	ASMN_MENU						
String2	NONE						
String3	NONE						
String4	NONE						
Installation Data Link Name	NONE						
SCHED created							
Window UTILITY/1 at DELL MCP							

Then Define a new MCS window with Supervisor as the MCS and your new Installation Data.

EX.

W - WINDOW ACTIVITY (MCS)		COMS					
Action:	CR	MODIFY	INQUIRE	DELETE	GO	HOME	(Press SPCFY for help)
	SE	FI	LA	NE	PR		
Window Name	SCHED						
Subaddress							
Window Type	M - Direct R - Remote-File M - MCS						
Virtual Terminal Name	DEFAULT						
Maximum Users	0						
Notify Open (Y/N) and Text	N						
Notify On (Y/N) and Text	N						
Installation Data Name	SCHED						
MCS Window specific fields:							
Truncated Results (Y/N)	N						
Hostname or Domain Name	NONE						
MCS Title	METALOGIC/SUPERVISOR						
Remote Window Name							
Single Window (Y/N)	N						
SCHED created							
Window UTILITY/1 at DELL MCP							

With this setup ?ON SCHED will open a Supervisor window (without checking Supervisor Window Security). It will then DO ASMN_MENU (or whatever ODTs you specified in the Installation Data). The ODTs will communicate using SHOW, to

send to the window, and INPUT, to get info from the window. Any data entered when there is no Input function waiting will be ignored and "Station busy" will be sent to the last line of the window.

The intent of this change is to give more precisely controlled access to Supervisor by allowing the defined ODTs to filter the commands allowed.

From a REMOTESPO

For installations without COMS, or which do not wish to use a dedicated window, a more primitive remote file interface is available, called the REMOTESPO. It is activated by a REMOTESPO <station> command. By default, only one REMOTESPO is allowed active at one time, but this can be changed by META/INSTALL and the SUP_MAXWINDOWS configuration variable. For details see the above section on SUPERVISOR COMS WINDOW. SUPERVISOR's REMOTESPO is intended for use on hardcopy or TD830-like terminals only and is subject to an appropriate NDL driving the stations. (Other terminal types may or may not work depending on the NDL.)

If your terminal is attached to the system via COMS, it makes a difference whether the <station name> you specify in the REMOTESPO command is the actual station name (the name known to COMS) or the station name of the CANDE window (the name known to CANDE). If you use the actual station name, the REMOTESPO will be opened in a new COMS window named REM0001 – or the 0001 may be a larger number if some remote files are already open at your station.

If you use the station name of the CANDE window, or if your terminal is connected directly to CANDE, the REMOTESPO will be opened through a remote file in your CANDE session. This may cause CANDE to announce the remote file open to the user and/or ask the user to **RESPOND OK OR DENY** to the file open, depending on what value the System Administrator has assigned to the CANDE option LAISSEZFILE at your site.

MCSes

An MCS can input commands to SUPERVISOR, either directly, or if it provides an RJE-like interface to input system commands. SYSTEM/RJE provides this feature, but CANDE and COMS edit all system commands, and do not forward unrecognised commands to the system CONTROLLER.

DCKEYIN Intrinsic

SUPERVISOR responds to programs that use the DCKEYIN DCALGOL intrinsic. A number of commands are restricted and will not be accepted from DCKEYIN programs, in particular those that assume the type of full screen paging used on ODTs and REMOTESPOs. Note that the only response returned by DCKEYIN is "TEST FUNCTION".

EXTERNALCOMMAND

For advanced interactivity with Supervisor the EXTERNALCOMMAND entrypoint may be called to pass commands or queries.

Example Program

```
00000100BEGIN LIBRARY SUP(LIBACCESS=BYTITLE,
00000150          TITLE="*METALOGIC/SUPERVISOR.");
00000200  INTEGER PROCEDURE EXTERNALCOMMAND(INP,OUT);
00000300  ARRAY INP,OUT[0];
00000400    LIBRARY SUP;
00000450  POINTER P;
00000500  ARRAY INP,OUT[0:299];          REAL T,Z;
00000600  FILE RMT(KIND=REMOTE,MYUSE=IO,MAXRECSIZE=80,BLOCKSIZE=80,
00000610          FRAMESIZE=8);
00000630  %% ASSUMES THAT SUPERVISOR IS ON DISK
00000660  REPLACE MYSELF.FAMILY BY ".";
00000700  REPLACE INP BY MYSELF.TASKSTRING;
00000800  T:=EXTERNALCOMMAND(INP,OUT);
00000820  P:=POINTER(OUT);
00000840  WHILE T GTR 0 DO
00000860  BEGIN
00000900    SCAN P FOR Z:T WHILE NEQ 48"0D";
00000920    WRITE(RMT,T-Z,P);
00000940    P:=*+(T-Z+1);
00000960    T:=Z-1;
00000980  END;
00001000END.
```

It is meant to be run from CANDE or MARC; you can pass any SUPERVISOR command into the program using TASKSTRING. If the program runs under the user IPP, then you can use FOR IPP without any password to lock activities or execute Opals.

The ExternalCommand interface to Supervisor requires the caller to be running from a privileged or system user and for the codefile calling ExternalCommand to be MPed PU or Taking.

An alternative is to MP the codefile with SECADMIN capability. If set, SUPERVISOR does not require the usercode of the caller to be PU or SYSTEMUSER, allowing callers which have no associated usercode.

If MP PU or MP TASKING is set and SECADMIN is reset, EXTERNALCOMMAND will continue to require a usercode with the appropriate privilege.

If the calling task is MPed SECADMIN then no check is made on the calling usercode otherwise if the calling usercode is not PU or SYSTEMUSER the task will be dsed.

Any task using EXTERNALCOMMAND which is MPed SECADMIN should be carefully controlled as it could give unrestricted ODT access.

If the Supervisor logging option is enabled then an 'Err' entry will be made in the Supervisor log giving the mix number of the task which attempted an unauthorised call on ExternalCommand with the reason that it failed.

For example:

```
RUN$OBJECT/EXT;TASKSTRING="FOR IPP AF + 1200:TT DO TEST"
#RUNNING 16007
#?
SCHEDULED IN 1 TODAY SLOT
#ET=0.1 PT=0.0 IO=0.0

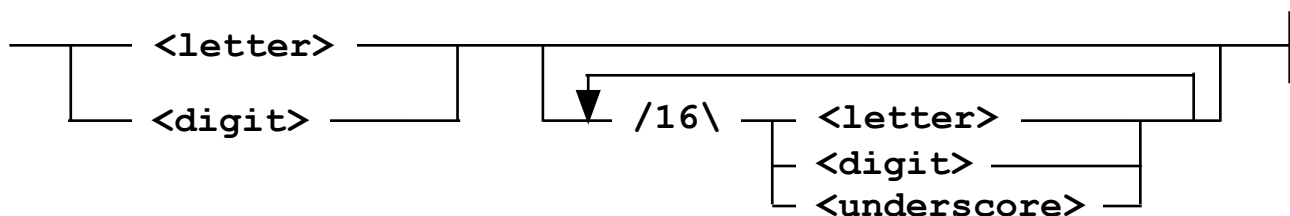
RUN$OBJECT/EXT;TASKSTRING="AF ? 1200"
#RUNNING 16009
#?
----- PENDING SCHEDULE after 12:00 TODAY -----
1: FOR (IPP) TT DO TEST
#ET=0.0 PT=0.0 IO=0.0
```

The program will return multi-line responses for interrogations etc.

HARDCOPY Interface

SUPERVISOR has a complete interface to the Unisys HARDCOPY program. In particular, inputs from SUPERVISOR's REMOTESPO, SM messages and commands from sources that generate no response are all sent to HARDCOPY, including the (normally suppressed) response. In general, any input to SUPERVISOR will appear prefixed with the text "SUPERVISOR". ODT input via TT is logged by CONTROLLER and is never logged by SUPERVISOR.

OPAL Identifiers



OPAL identifiers are similar to file identifiers in WFL. They can start with a letter or digit, and can have embedded underscore characters "_" – e.g. ANY_OLD_ID_1. This identifier is then stored as the name of the program and is used to list or print it.

If a FOR clause precedes the DEF command, or if the terminal is running with a TERM USER specification, the identifier is entered into SUPERVISOR with a

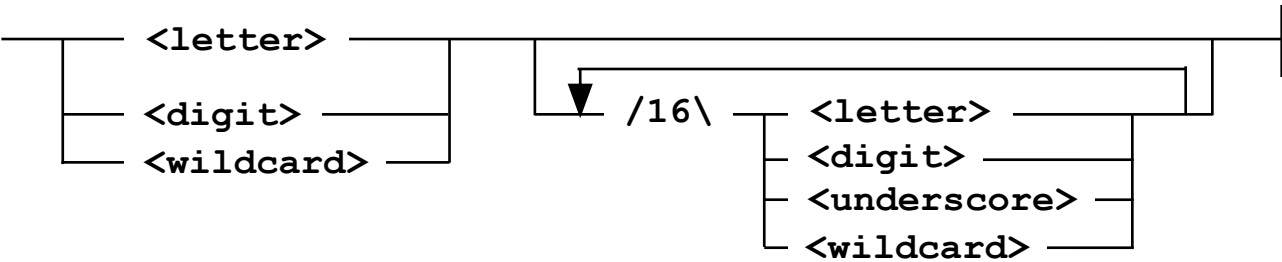
usercode attached. This OPAL program can then only be run, listed or deleted by that usercode (or a privileged usercode if the creating usercode is not privileged.)

Examples

```
CPU_TIMES
SHOW_BARS
12_TILL_2_SHIFT
(OPER) USERDATA_SV
```

Wildcards and Patterns

<pattern>



In many SUPERVISOR commands, it is possible to limit enquiries by using "wildcards" to DEFINE a pattern for target OPAL identifier specifications. For example, to specify all OPAL programs starting with "TOUR", you would use the wildcard form:

```
TT DEF ? DEF TOUR=
```

The available wildcards are as follows:

Wildcard	Action
"="	match any string, including an empty string
"?"	match any single character
"&"	match any alphanumeric character (upper and lower case letters and digits)
"@"	match any upper or lower case alphabetic character
"#"	match any numeric digit
"~"	match any string not including "/"

Wildcard	Action
"\"	Escape character use before any other wild card to use the literal value of the character not its wildcard action. Ex. =A\=B= would match any string containing A=B

Examples

```
TT HELP ATT =PROC=
```

returns all ATtributes with the literal "PROC" in the name

```
TT DEF ? SITU JOB####=
```

will display a list of all SITUations whose names begin with "JOB" followed by four numeric digits.

```
TT DEF ? ODTS ?=#####@_RUN&&
```

will match the ODTSequence named "1993125B_RUN0A"

The wildcard facility enhances many SUPERVISOR commands with considerable flexibility and searching power. Those commands which can accept wildcards will have <pattern> shown in their syntax railroad diagrams.

Repeat Last Output (EH) Command

—— EH —————|

When a command is given to SUPERVISOR, the response comes back to the originating terminal. However, unless it is a multi-screen response, the output is not synchronised with CONTROLLER. What can sometimes happen on an ODT is that a CONTROLLER output (such as an ADM screen on an ODT) can overwrite something that has just come from SUPERVISOR. If this happens, the EH command will re-display the last screen output by SUPERVISOR.

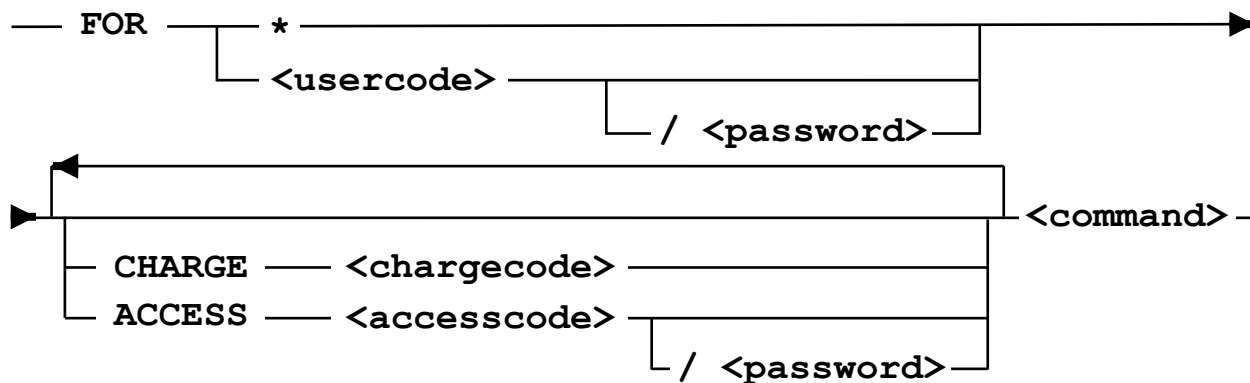
Example

```
TT EH
```

See Also:

[TERM](#) Command

The FOR Modifier



FOR is not a SUPERVISOR command in its own right; rather, it is a modifier to other commands and allows a USERCODE< ACCESSCODE or CHARGECODE to be added to them. SUPERVISOR checks that the usercode and password are valid before processing the command. Note that SUPERVISOR does NOT store the password, it merely verifies that it is valid in the USERDATAFILE.

For AFTER, WHEN and DO commands, using FOR has the effect of ‘locking’ the activity from being deactivated or removed. To cancel the locking, the original locking usercode and password must be used. See FOR with AFTER etc. later in this section.

Some examples:

```
TT FOR META/PASSWORD TT DO MYOPAL
TT FOR TEST/TEST ENTER FROM *OPAL/FILES ON WORK
TT FOR PRIV/PU TT WHEN SECURITY DO SECURITY
```

Note that, if the '*' modifier is used, it has the effect of removing an assumed usercode. For example, if the command is given from a terminal with a usercode attached via a TERM USER command, the usercode can be suppressed by using a 'FOR *' modifier.

From a Supervisor window you may use the logged on usercode without a password. So if the Supervisor window was entered from a session logged on as BOB then FOR BOB AE... would be valid.

The FOR modifier may also be used to invoke WFL commands directly from a SUPERVISOR window or within an ODTSEQUENCE:

```
TT FOR META/PASSWORD ACCESS TEST/TEST START WFL/TEST ON DEV
TT FOR META/PW ?DISPLAY MYJOB(USERCODE)
TT FOR META CHARGE C1234 RUN *BATCH/APPLICATION
```

Password handling with NODEFAULTUSE

Supervisor has an alternative facility that allows the FOR modifier to be used *WITHOUT* supplying a password, but only under certain controlled conditions. The new USERDATA attribute NODEFAULTUSE is widely used by Unisys' SYSTEM/ASSISTANT to control those usercodes which can be used to invoke WFL jobs without knowing passwords.

If the Supervisor USE option NOPASSWORDS is not set, the normal security enforcement of password validation remains in force. However, if USE NOPASSWORDS is set, Supervisor will permit the "locking" of activities under usercodes, without password validation, as long as the locked usercode itself does NOT have the NODEFAULTUSE attribute set.

It is not possible to lock an activity under the Supervisor usercode without a password at any time.

If USE NOPASSWORDS is set, commands such as:

```
TT FOR META AFTER+1400:TT DO MY_ODTS
TT FOR META WHEN MSG DO MSG
```

The above commands are permitted from any Supervisor window, ODT or active ODTSequence as long as the example META usercode also does **not** have the NODEFAULTUSE attribute assigned.

It is STRONGLY RECOMMENDED that if USE NOPASSWORDS is set, all usercodes that require the password validation with a FOR modifier MUST have NODEFAULTUSE assigned by the Security Administrator.

The following command can be used to identify user codes which will be at risk if TT USE NOPASSWORDS is set:

```
TT FOR <secadmin user>/<password> EV (USER:NOT NODEFAULTUSE) DISP (USER)
```

The following table may help to clarify this feature; the Supervisor usercode is assumed to be "SUPER":

FOR usercode	USE NOPASSWORDS	NODEFAULTUSE set on usercode?	Is FOR with no password allowed?
META	Yes	Yes	No
META	Yes	No	No
META	No	Yes	No
META	No	No	*YES*
SUPER	Yes	Not applicable	No
SUPER	No	Not applicable	No

Multiple FORs

If the password for a usercode is not available, a more privileged usercode (PU) may use two FOR modifiers, the first with the more privileged usercode and password, and the second with the less privileged and no password. This allows a privileged usercode (PU) to act as a non-PU without giving its password. To do the same for a PU, the usercode of the ODTSECURITY user must be used, e.g.

```
FOR PROG/ALGOL ...
FOR PU/PW FOR PROG ...
FOR SECAD/REALLYSECURE FOR PU ...
```

The **FOR** modifier only has an effect on the following commands:

FOR with AFTER

If an activity is scheduled with a **FOR** modifier, **SUPERVISOR** will check that the usercode, and the associated password if supplied, is a valid usercode. The usercode, but not the password, is retained with the scheduled activity, and when the activity is performed, the usercode is checked again. If valid, the command is executed; in the case of system commands with the usercode in the same way as the session usercode is passed with a control ("?",) command from **CANDE**. All jobs started this way will run under the usercode specified at time of entry. If the usercode is found to be no longer valid the activity is not performed, and the postscript of the activity is flagged as "INVALID USER".

Deletion of a scheduled activity with an associated usercode will only be accepted if a **FOR** modifier with the same usercode (and any password matching the usercode) is supplied. The password need not be supplied if a **FOR** modifier with a more privileged usercode is supplied, as above.

FOR with DEF

FOR is used to add a usercode to an **OPAL** program identifier. This allows a program to be locked under a usercode. It can then only be listed, deleted, modified or run if that usercode is supplied. Usercoded programs have the usercode in parentheses before their identifier in the **OPAL** directory lists.

FOR with DO

The **FOR** modifier is required to run any **OPAL** program that is 'locked' under a usercode. This has the side effect of also associating that usercode to any **ODT** statements that are performed by the **ODTSequence**, including any **WFL** jobs that may be subsequently invoked.

FOR with PRINT

As with the **DEFINE** command, a usercode is needed to send listings of usercoded **OPALs** to a printer-backup file.

FOR with USE

The variant of the **USE** command to change the usercode of the tape librarian or **SUPERVISOR's** usercode, for example:

```
TT FOR PU/PW USE USER SUPERVISOR
TT FOR PU/PW USE TAPELIB USER ANYU
```

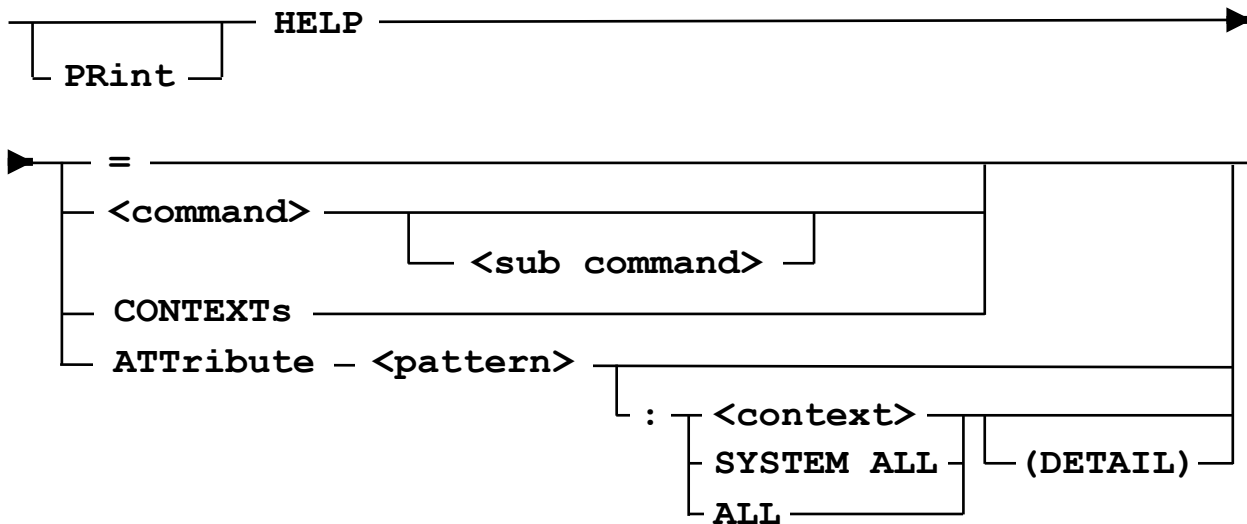
must have the **FOR** modifier of a **PU**.

FOR with WHEN

The FOR modifier may be used to protect any OPAL, invoked using WHEN or DO, that is required to be 'locked' under a usercode. This has the side effect of also associating that usercode to any ODT statements that are performed by the ODTSequence, including any WFL jobs that may be subsequently invoked.

A FOR modifier has no effect on a WHEN ? display.

HELP Command



HELP provides information about SUPERVISOR commands. Information is returned as one or more screens of explanatory text to the requesting terminal. **<command>** here means a SUPERVISOR command, and **<sub-command>** is normally a keyword in the syntax diagram of the appropriate command. If no command is specified, an index page of available topics is displayed.

HELP also gives information about OPAL ATtributes, or sets of ATtributes. A single ATtribute can be specified by using the **HELP ATT <id>** syntax where **<id>** is a unique OPAL identifier. If a **<pattern>** is used, all ATtributes which match the wildcard are listed in an abbreviated list.

If the HELP ATT is followed by the " : **<context>**", the command will only return attributes within the specified context. If no pattern is specified in this case then = is assumed. The command **HELP CONTEXT** gives information about the contexts currently available.

If the (DETAIL) options is used then full attribute information is returned for a list of attributes, instead of just names.

The output from HELP can be directed to a printer by using the command prefix **PRint**. The output device will be that specified by the USE DESTINATION command.

Examples

```
TT HELP
TT PR HELP WHEN
TT HELP PR ODTS
TT HELP ATT IOTIME
TT HELP CONTEXT
TT HELP ATT ELAPSED=:MX
```

Next Screen (NS) Command

Often, SUPERVISOR responses will occupy more than one screen at a time. Like CONTROLLER, multi-screen responses are queued for output and each next page is requested by using the Next Screen command:

The CONTROLLER and SUPERVISOR Next Screen Commands function similarly, causing the next page of a multi-page response to be returned. They also both have the side effect of temporarily stopping the ADM on an ODT.

On a Window Station SUPERVISOR will display and accept the **TT +** variant of the command and it may also be used to scroll forward after a **TT -** (Previous page) command has been input.

If the command that generated a multi page response was entered from a Window Station and was processed by CONTROLLER the original command line will be restored after the last page of the response has been requested.

Examples:

```
TT NS
TT +
```

Note that if no subsequent screen exists, SUPERVISOR will respond:

```
NO NEXT SCREEN SCANNING NS
```

or if the **TT +** variant has been input:

```
There is no Next Screen
```

Previous Screen (-) Command

On a Window Station SUPERVISOR the **TT -** (Previous Screen) command may be used to review screens that have been displayed previously.

Scrolling backwards and forwards through screens can be achieved by using both the **TT +** and **TT -** commands.

Examples:

```
TT -
```

Note that if no previous screen exists, SUPERVISOR will respond:

There is no Previous Screen

PRint Modifier

Sometimes it is convenient to produce a hardcopy of the displayed output from SUPERVISOR. Any SUPERVISOR command may be prefixed with PRINT or PR to direct the output to the printer.

The syntax for redirecting output from the screen to the printer is:

—— **PRint** ———— **<SUPERVISOR command>** ———— |

The printout will be directed to the DESTINATION specified by the SUPERVISOR USE command. Printouts are produced under the USE-specified usercode (default = SUPERVISOR) and with a "dummy" chargecode of PR/CMD (for PRint command). For example, to print the output from the HELP command enter:

TT PR HELP EH

To print the entire HELP file simply enter

TT PRINT HELP =

SUPERVISOR will respond with "REPORT PRINTED" when the report has been generated.

VIA Modifier

VIA is used as a modifier to any other SUPERVISOR command. With a few minor exceptions, it treats the command as if it had been entered from the nominated ODT (system console) and returns any response to that ODT. **<ODT number>** is an integer unit number representing an ODT. The **<SUPERVISOR command>** can be any SUPERVISOR command although this re-direction is most useful in the invocation and disabling of WHENs, especially if linked to DISPlays.

Note, however, that this syntax only applies to ODTs and cannot be used to re-direct output to REMOTESPOs.

—— **VIA** ———— **<ODT number>** —— **<SUPERVISOR command>** ———— |

For example, from any REMOTESPO or ODT:

TT VIA 1 WHEN TEST_SITU DISPLAY TEST_DISP

would invoke the above WHEN as normal, but it would appear to come from ODT/1, as if in response to a **WHEN ?** command. Similarly, from any ODT or REMOTESPO:

TT VIA 1 WHEN TEST_SITU DISP

would turn it off.

TT VIA can be useful in re-directing a label-printing DISPlay to an ODT that has a

label printer attached. Also, if an OPAL program has previously been running on an ODT which becomes unavailable after a system reconfiguration, the **TT VIA** option is one way to "turn off" the OPAL (which SUPERVISOR will have restarted after the system switch).

A tip to find the number of any ODT is to enter the following at the desired console:

```
LABEL SUPER
```

followed by

```
PER SC
```

The response to the **PER** command will indicate which ODT the command was entered from.

```
PER SC
```

```
----- SC STATUS -----  
233      S C R A T C H  
234      SUPER  
235      S C R A T C H
```

Subsequently entering:

```
CL SC 234
```

will remove the label.

Exceptions

If **<ODT number>** is not a valid ODT, the command is treated as if it was entered from the highest numbered unassigned ODT.

If the command is not entered using **<mix no>SM** then any error messages or simple responses will be sent to the originating station.

Control Commands

This chapter deals with those SUPERVISOR commands that control the running of the software and its environment, or which return information about the software. In this chapter (and those following), each command is dealt with in a similar fashion. First, the syntax of the command is DEFINEd using standard Unisys Railroad Diagrams. It is assumed the reader is familiar with this method of defining command syntax.

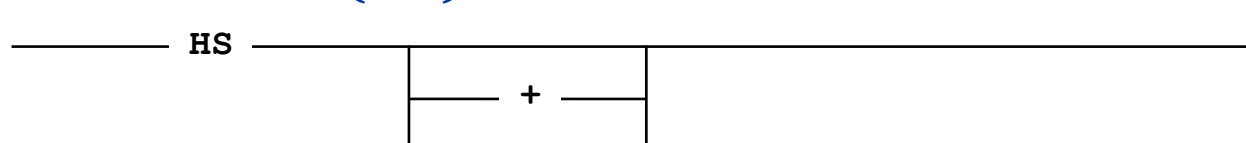
Next, the command is discussed in detail and examples given to show how it is used in practical applications. Lastly, any related commands are listed for easy cross-reference.

Some of these commands will already have been introduced in [Getting Started](#) and the reader can refer back for more examples.

Commands dealt with in this section are:

Commands	Synonym	With After
HS	–	No
QUIT	–	Yes
REC	-	Yes
REMOTESPO	–	Yes
SO	–	Yes
SS	–	Yes
TERM	–	No
TP	–	Yes
USE	–	No
WINDOW	–	Yes
WS	–	No

Hold Schedule (HS) Command



The HS command interrogates (**HS**), sets (**HS+**) or resets (**HS-**) the SUPERVISOR Hold Schedule flag. This should not be confused with the normal HS system command (which holds the system schedule). To use the system HS command from a Supervisor window the command must be prefixed by ODT.

The SUPERVISOR HS flag allows for cases where, for a period of time, performing some of the scheduled activities is not required, but it is inconvenient to delete them from the schedule. Unlike the earlier HS implementation, the flag will remain in force across H/Ls and restarts of SUPERVISOR.

```
TT HS +
```

```
SUPERVISOR SCHEDULE IS NOW HELD
```

When the Hold Schedule flag has been previously set by **HS+**, each scheduled activity that reaches it's appointed time is automatically marked as "suspended" and will be skipped. A **TT AF ?** will show:

```
TT AF ? 0920-1000
```

```
----- SUSPENDED SCHEDULE after 09:46 TODAY -----  
1: DISPLAY("THIS IS A TEST")
```

If an **HS+** is active over midnight, SUPERVISOR will load the new days schedule as normal but an "image" of the previous day is preserved in the SCHEDULE file until an **HS-** is issued.

Using an **HS-** command will immediately resume the schedule; then SUPERVISOR will "back track" the current schedule to the time that the original schedule was held, reloading the previous days schedule if needed.

```
TT HS -
```

```
SUPERVISOR IS NO LONGER HOLDING ITS SCHEDULE
```

Once resumed, SUPERVISOR invokes a special task called BACKTRACK/ SCHEDULE to handle the rollback of the SCHEDULE to the previous day and begin processing missed activities. At this point, the mode of the back track operation depends on the setting of the USE MODE ... FOR BACKTRACK (see **USE** command in this chapter for more details). Please note that the old SUPERVISOR option, AUTOBACKTRACK, has now been replaced by this mechanism.

If **BACKTRACK** mode is **AUTO**, then SUPERVISOR will automatically re-process all missed activities, starting with the remaining schedule of the day preserved from the time of the initial **HS+**. This is then followed by a reload of the current days schedule, if required. Any intervening day schedules will NOT be processed; this applies regardless of the setting of **BACKTRACK**.

If **BACKTRACK** is **NONE**, then SUPERVISOR will automatically discard all missed activities from all schedules. **TT AF ?** will show these discarded activities marked as "skipped".

If **BACKTRACK** is **TODAY** then SUPERVISOR will only process the missed actions from the current day's schedule; any earlier schedule will be loaded and processed but any missed activities will be skipped.

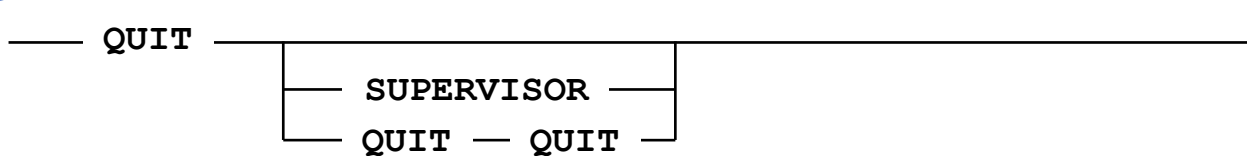
Lastly, if **BACKTRACK** is **VERIFY**, then SUPERVISOR requires operator intervention to confirm the processing of each missed activity. The interface provided works most effectively from an ODT but interaction is also possible from a SUPERVISOR COMS window. Each command requires a **TT SKIP** or **TT OK** to, respectively, reject or allow the activity to be processed. **VERIFY** mode is discussed in much greater details in the **USE** command section.

Previously, **HS+** would not affect certain scheduled Supervisor commands such as **SAVE SCHEDULE** or certain **PRINT** commands. Now, all scheduled SUPERVISOR commands will be held without exception. Further, if the **HS+** flag is set during the event of a Halt-Load, SUPERVISOR will not invoke the AFTER HALTLOAD recovery process.

See Also:

[USE](#) command (MODE modifier)

QUIT Command



The QUIT command is normally entered on a SUPERVISOR window to end that session. However, the two variants are used to force an immediate and orderly termination of SUPERVISOR. If the **QUIT QUIT** variant is used, SUPERVISOR must be restarted manually.

The **QUIT QUIT** variant is only valid from a real ODT or via an SM command.

If the **SUPERVISOR** variant is used, it will restart automatically. SUPERVISOR will reject a QUIT command if any processed utilities (FILEDATA, LOGANALYZER, FAMILYMANAGER, etc.) or JAMPACK are running.

The sequence of operations after a **QUIT SUPERVISOR** is:

1. Any of SUPERVISOR's subtasks, if running, will be terminated or DS-ed.
2. All WHENs running will be stopped, and their sessions will be closed and logged.
3. If, and only if, the QUIT SUPERVISOR variant is used, SUPERVISOR will ensure its own restart by a ??RUN of its code file.
4. The current SUPERVISOR stack will go to end-of-task.

When SUPERVISOR restarts, all WHENs (except those that were "**WHEN . . . DISPLAYs**" and were started from SUPERVISOR windows or REMOTESPOs) will be automatically restarted.

The QUIT command may be the subject of an AFTER command.

Example

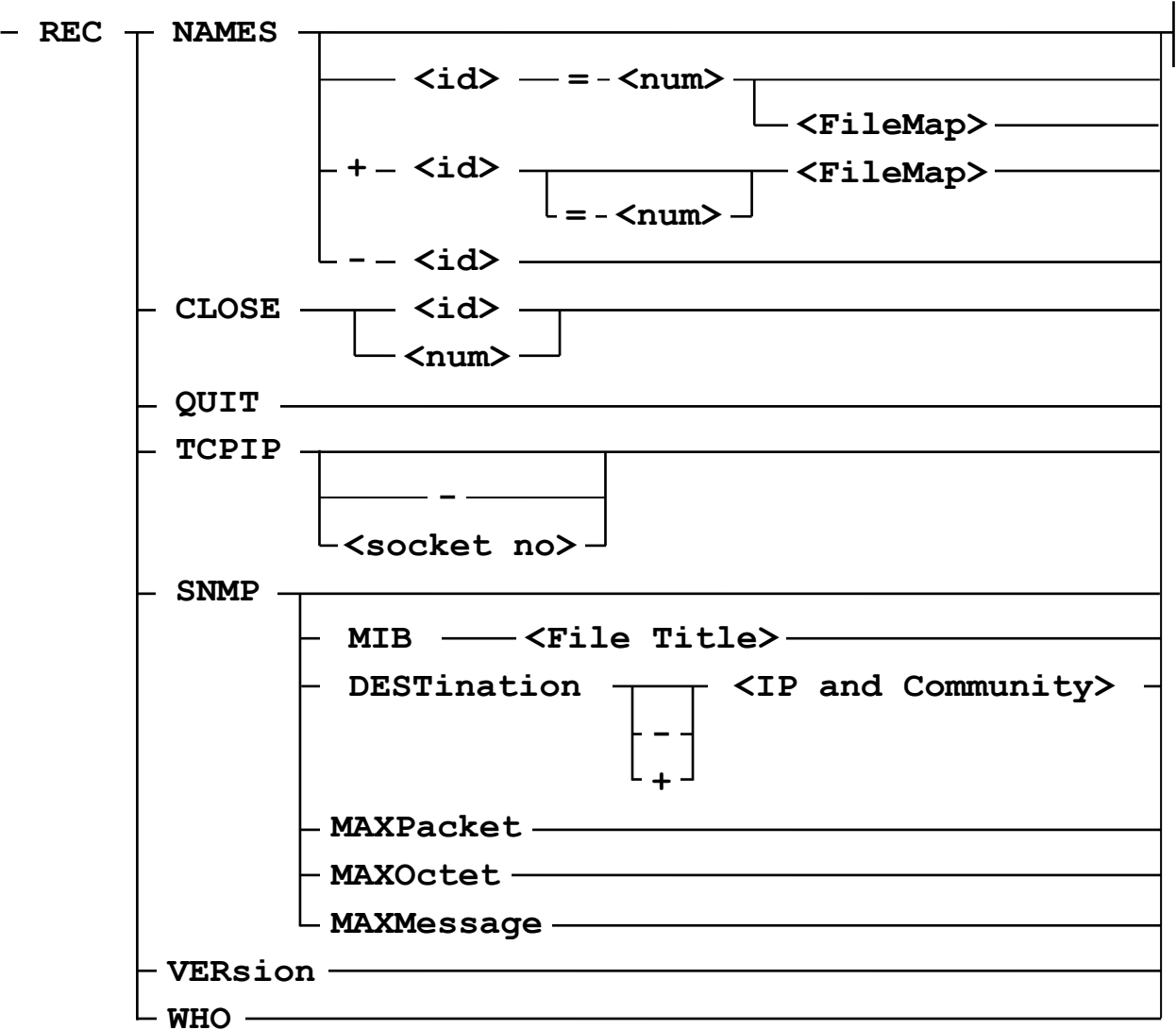
AF 0300 DAILY TT QUIT SUPERVISOR

REC Command

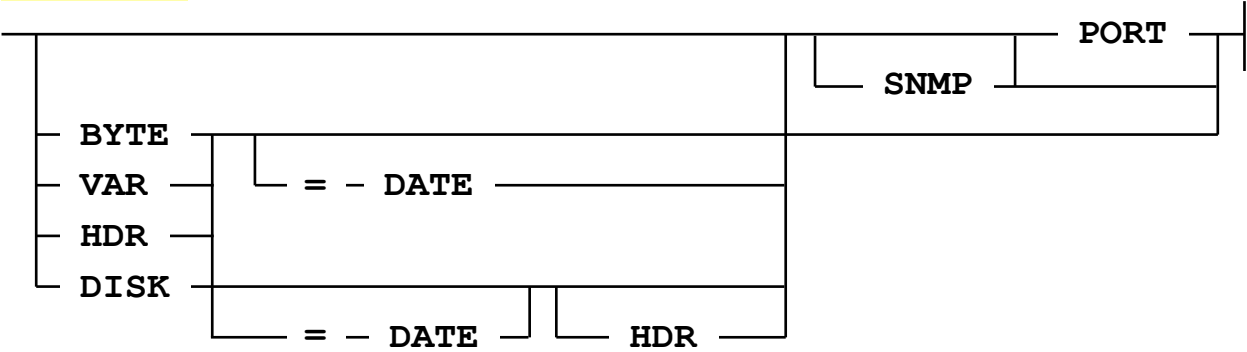
The REC command permits direct communication with the Metalogic RECORDER program. RECORDER passes information from active OPAL programs that use the RECORD statement, to HOTLINE clients or direct to DISK files.

Please see [RECORDER and HOTLINE](#) for more information.

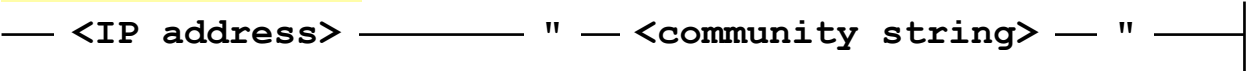
The full command syntax is shown below:



<FileMap>



<IP and Community>



Names

The **NAMES** variant allows the assignment of meaningful mnemonics to RECORDER action values, which can then be used in any RECORD statements instead of integer values. Examples of the use of each form of this command are listed below:

TT REC NAMES

The unqualified form returns details of any established mappings, a typical response might be:

REC NAMES

```
--- RECORDER File Mapping Mnemonics ---
--- 0 of 18 Disk Files currently open ---
RECORD [12] mapped to BYTE          BYTE
RECORD [13] mapped to VAR           BYTE
RECORD [14] mapped to VOLS          BYTE
RECORD [15] mapped to RESP          DISK
RECORD [16] mapped to TAPEREREPORT  BYTE
RECORD [17] mapped to DEBUG         PORT
RECORD [18] mapped to BOB           BYTE
RECORD [19] mapped to STATS         BYTE
RECORD [20] mapped to RN            PORT
RECORD [21] mapped to CFGSAVE       BYTE
RECORD [22] mapped to HEROIX        PORT
RECORD [23] mapped to LOGEOJ        BYTE
RECORD [24] mapped to META_STATUS   PORT
RECORD [27] mapped to META          PORT
RECORD [28] mapped to RSVP          PORT
RECORD [29] mapped to INQUIRY       DISK
RECORD [31] mapped to BN            DISK
RECORD [32] mapped to REJECTED      PORT
RECORD [33] mapped to AC            DISK, PORT
RECORD [34] mapped to TH            DISK
RECORD [35] mapped to META_RECORD   PORT
RECORD [36] mapped to SNMP          SNMP
RECORD [37] mapped to META_ALERT    PORT
```

```
Default DISK files NOT mapped 0,1,2,3,4,5,11
Default PORT files NOT mapped 6,7,8,9,10,11,47
Default VAR files NOT mapped 50,51,52,53
```

The TT REC NAMES command also enables mnemonics to be mapped to specific file numbers. New or existing file mappings can be mapped to user-selected file numbers as long as the number is not already mapped.

For example:

```
TT REC NAMES RSVP = 3 PORT
TT REC NAMES + TEST = 25 BYTE
```

In the first example, the existing mnemonic RSVP is being re-mapped to file number 3 and is assigned a kind of PORT; any existing assignment for RSVP is deleted and its prior file number is returned.

In the second example shown above, the new mapping TEST is associated with file number 25 and is assigned a kind of BYTE.

A further variant of the TT REC NAMES command allows a mnemonic mapping to

be cancelled, for example:

```
TT REC NAMES - TEST
```

would remove the mapping of the mnemonic TEST

By default any DISK files created by a RECORD[<mnemonic>] statement in OPAL will be created on the DL BACKUP family with the title:

```
*SUPERVISOR/RECORD/<mnemonic>/<dayname>
```

or for BYTE files:

```
*SUPERVISOR/RECORD/<mnemonic>/"<dayname>.TXT"
```

If the = DATE modifier was used <dayname> is replaced by the current date in yyymmdd format.

For example:

```
*SUPERVISOR/RECORD/BOB/"20080325.TXT"
```

The action taken by RECORDER when a <File Map> specification has been assigned is as follows:

PORT

Will cause RECORDER to pass the message to any HOTLINE program requesting this message type.

SNMP

Will create an SNMP Trap which will be passed to any IP address specified by the **REC SNMP DEST** command. The **REC SNMP MIB** command can be used to generate a MIB file to match any traps specified.

DISK

Means that messages will be written to a JOBSYMBOL file (this is the same format as **RECORD [1]**). If **DISK** is further qualified by **HDR**, or **HDR** is specified on its own, it will result in messages being written to a JOBSYMBOL file, with each message preceded by a header line. This is the same format as for **RECORD [0]**.

VAR

Means that messages will be written to a variable length record data file. This is the same format as **RECORD [50]**.

BYTE

Will cause RECORDER to write messages to an ASCII byte stream file, suitable for reading via NX/Services.

CLOSE

The **REC CLOSE** variant immediately closes the RECORD file specified by the mnemonic **<Id>** or the Record File number **<File No>**. This allows the file to be viewed off-line on a PC (if it is BYTE format and available on a mapped MCP share) or directly from the machine.

QUIT

The **QUIT** variant provides a convenient way to initiate an immediate closedown of the RECORDER program.

TCPIP

The **TCPIP** variant enables the user to control the communication service (TCPIP or BNAv2) used for the RECORDER Port files. The current status is always shown in the response to a **TT REC VER** command as detailed below. Typical commands might be:

```
TT REC TCPIP -
```

or

```
TT REC TCPIP 22222
```

Using the **' - '** modifier will cancel a TCPIP assignment and cause RECORDER to switch to a BNAv2 setting. Using the **<Socket No>** modifier will cause RECORDER to use TCPIP, with socket number assigned by **<Socket No>** (in this example 22222) instead of BNAv2. Any change to the transport service will only be applied after the next RECORDER restart.

Using the mapping facility, RECORDER now supports a specific TCPIP port file to a nominated IP address.

```
REC NAME + HEROIX PORT=10.0.0.26
```

A **REC QUIT** should be done after setting or changing any PORT mappings.

Record messages passed to this port have no additional data added. (i.e. Not timestamp or Hostname). The port is closed after each write.

This change was made to enable Supervisor to create HEROIX ROBOEDA events. The layout of the events is controlled by Opal routines. Metalogic have a suite of example Opals available to anyone interested in passing events to the ROBOEDA application. The **REC TCPIP** command is used to specify the port number to be used (2919 is the default value for ROBOEDA).

SNMP

The SNMP variant is used to control SNMP traps sent by Recorder when a RECORD statement, which maps onto type SNMP, is performed. SNMP traps do not require UNISYS SNMP to be configured.

Five options can follow the SNMP variant. If no option is specified then the SNMP settings are reported.

MIB

The MIB option creates an SNMP Mib file to match any Name Mnemonic set to send an SNMP trap. Ensure that the title passed has a Usercode and Family and that any level with a dot character is quoted. The generated file should be transferred to any system used to monitor SNMP traps, and be compiled into the MIB database.

Note that if MAXOctet or MAXPacket are changed then the MIB file should be regenerated as the number of variables in a Trap could change.

The MIB file contains extensions which can be used by the Tivoli mib2trap utility.

Example

```
REC SNMP MIB (BOB) SNMP/"META.MIB" ON DEV
```

DESTination

The DESTination option is to add an address and community string where Traps should be sent. If there are no addresses specified then no SNMP TRAPS can be sent. To delete a specific Ip/Community the - prefix is used and both IP and community must be specified.

Examples

```
REC SNMP DEST 10.0.0.35 "supervisor"  
REC SNMP DEST - 10.0.0.1 "public"
```

MAXPacket

The MAXPacket option is to specify the maximum size of UDP packet that can be handled on your SNMP client systems. It defaults to 1500 bytes which should work for most ethernet connections.

Messages which would create packets larger than this are spit into multiple traps. The intent of this setting is to allow you to avoid fragmented UDP packets. Some systems will correctly reassemble fragmented UDP packets, others do not. On systems which do not reassemble packets even the first part will usually be ignored.

MAXOctet

The MAXOctet option is the maximum size of octet string which will be generated. It defaults to 512 bytes. Some SNMP trap clients will not accept a trap containing an octet string > 512, some may not accept strings > 256. If a message is bigger than MAXOctet it is split into multiple octet strings.

MAXMessage

The MAXMessage option is the maximum size of text which will be sent for a single

Record. If this value is greater than 0 then messages will be truncated to this size before being split into packets.

VERsion

The **VERSION** modifier returns version and compile-time information about the RECORDER codefile. It will also display details of the Port in use if TCPIP communication is in use.

A typical response to a **REC VER** command might be:

REC VER

```

    --- RECORDER status ---
RECORDER Version 51.510.12
    Compiled 11:34:19,26/10/2007
Task is *METALOGIC/SUPERVISOR/RECORDER (5814)
TCPIP port file support is currently active
    MYNAME port number: 22222
No filters are running

```

WHO

The **WHO** modifier returns information about 'who' is connected to Recorder. The command is intended to help Metalogic in the diagnosis of Recorder problems.

Ex.

```

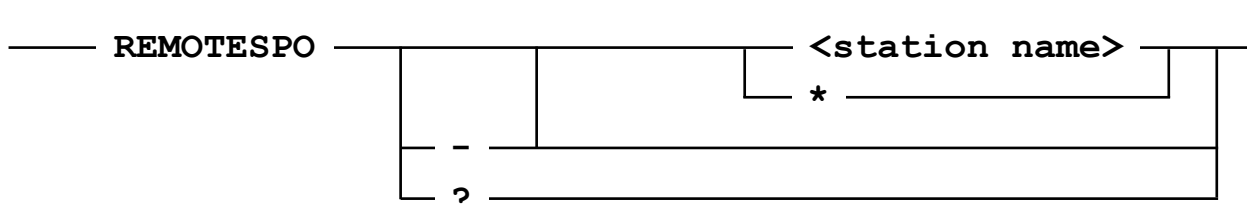
--- DELLMCP RECORDER CLIENTS ---
CL [00] LINKED to HOTLINEAGENT (00490), MASK=37,35,32,30,24,17
CL [01] LINKED to HOTLINEAGENT (00516), MASK=37,35,30,24,17
CL [02] NOT LINKED
CL [03] NOT LINKED
CL [04] NOT LINKED
CL [05] NOT LINKED
CL [06] NOT LINKED
CL [07] NOT LINKED
PORT[01] "127.0.0.1" , FORMATTED, AGED, MASK=46,20

```

Lines starting with CL show any client linked using the Connection Library interface.

Lines starting with Port show any clients linked using the Port File interface

REMOTESPO Command



Note: Use of this command is now discouraged. The recommended interface to Supervisor is the Supervisor COMS window.

The SUPERVISOR REMOTESPO is closely analogous to the system REMOTESPO command and has the same meaning. REMOTESPO followed by a **<station name>** causes SUPERVISOR to attempt to open a datacom file on the requested station and allow it to be used for the entry of system and SUPERVISOR commands.

REMOTESPO ?

lists all currently active REMOTESPOs.

REMOTESPO- <station name>

closes the REMOTESPO on the specified station.

REMOTESPO-

performs a **REMOTESPO- <station name>** for all active stations. This command is not permitted from a REMOTESPO.

REMOTESPO *

is a special case to allow a station to make itself a REMOTESPO, but the controlling MCS of the station must send a privileged usercode along with message.

REMOTESPO - *

is equivalent to using **?CLOSE**.

WARNING: Any REMOTESPO station has the same access to SUPERVISOR commands as does a system console and, in addition, SUPERVISOR will forward any non-SUPERVISOR commands to the system without limitation. Sites have the option of validating REMOTESPO input using METASECURITYLIB. This is detailed in Appendix A on Security.

NOTE: In default conditions, only one active REMOTESPO is permitted; this is because SUPERVISOR COMS windows are the preferred means of gaining interactive access to SUPERVISOR. As SUPERVISOR supports 16 current stations, this means that the maximum number of COMS windows is 15.

Setting the Magus configuration variable SUP_MAXWINDOW using the Metalogic INSTALL utility from CANDE can alter this behaviour:

U META/INSTALL CONFIG

And changing the value of SUP_MAXWINDOW by entering the Supervisor sub menu.

A restart of SUPERVISOR is required to enforce the new window configuration. It is strongly recommended that the number of REMOTESPOs be kept to an absolute minimum; the COMS windows access mechanism is a much more stable and reliable implementation.

Examples

```
TT REMOTESPO -  
TT REMOTESPO TSD001  
TT REMOTESPO ?
```

See Also:

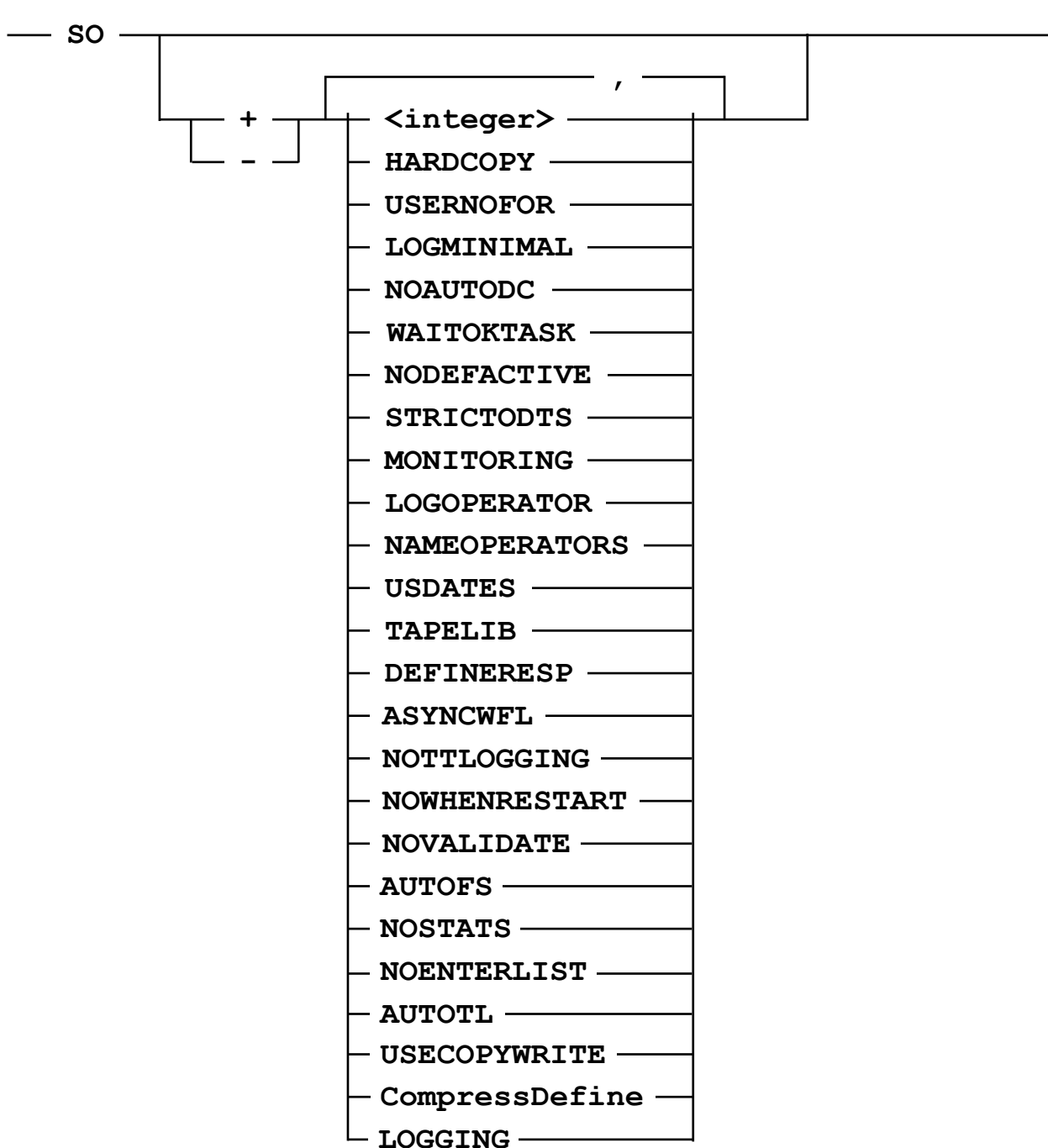
[TERM Command](#)

[WINDOW Command](#)

[Security](#)

Set Options (SO) Command

The user can choose different options while running SUPERVISOR. They are interrogated by "SO", set by "SO + <option list>", and reset by "SO - <option list>", where <option list> includes names or numbers.



A typical response to the so command is shown below:

```

SO
----- SUPERVISOR OPTIONS -----
1 HARDCOPY          2 USERNOFOR
3 LOGMINIMAL        4 NOAUTODC
5*WAITOKTASK        7 NODEFACTIVE
8 STRICTODTS        10 MONITORING
13 LOGOPERATOR      18 NAMEOPERATORS
19 USDATES          22*TAPELIB
28 NOTTLOGGING      30 DEFINERESP
31 ASYNCWFL         32 NOWHENRESTART
33 NOVALIDATE       35*AUTOFS
36 NOSTATS          37*NOENTERLIST
38 AUTOTL           45 USECOPYWRITE
46 COMPRESSDEFINE  47*LOGGING

```

The set and reset of these options is similar to that of the MCP OP command; the use of a number or option mnemonic is permitted.

To set options 5 and 30:

```
SO +5,DEFINERESP  
----- SUPERVISOR OPTIONS JUST SET -----  
5 WAITOKTASK          30 DEFINERESP
```

HARDCOPY (Option Number 1)

This option starts the standard SYSTEM/HARDCOPY program. Unisys' SYSTEM/HARDCOPY is normally run as the Computer Supervisor program (by using the CS system command) but this conflicts with SUPERVISOR which has to be the Computer Supervisor program. Setting this option keeps HARDCOPY in the mix without making it the CS. Even if HARDCOPY is DS-ed, SUPERVISOR will automatically restart it.

HARDCOPY is a useful aid when debugging ODTSequences because all SUPERVISOR generated SYSTEM commands and CONTROLLER's responses (if any) are written to the HARDCOPY file. When HARDCOPY is run by SUPERVISOR, its FAMILY is set to place the output files on the DL LOG family, and to look for PRINTCOPY on the SUPERVISOR USE SYSTEM family.

HARDCOPY is run under SUPERVISOR's USE USER usercode and all disk files are created under this usercode. For more information about HARDCOPY, refer to HARDCOPY/PRINTCOPY in the **Unisys System Software Support Reference Manual**.

See Also

[**MONITORING**](#) Option

USERNOFOR (Option Number 2)

This option is used to impose some control over the use of the USER object. If it is reset (the default setting), all OPALs using must be run under a FOR specifying the ODTSECURITY usercode. If **USERNOFOR** is set, no FOR is needed. Setting the option also requires a FOR with the ODTSECURITY usercode.

LOGMINIMAL (Option Number 3)

This option only has an effect if the **MONITORING** option is reset. In that case, sessions are not allocated to WHEN slots, and therefore actions of SITUations and ODTSequences are not logged.

NOAUTODC (Option Number 4)

If datacom is quit while SUPERVISOR is running, SUPERVISOR will lose MCS status and will attempt to regain it. If the system option AUTODC is reset this is not possible and so SUPERVISOR will set the system AUTODC option in an effort to

regain MCS status. If the SUPERVISOR option NOAUTODC is set then SUPERVISOR will not set the AUTODC option but will repeatedly display a message informing the operator that MCS status has been lost. This may prove useful when a site is changing networks and does not want to have AUDTODC set.

NOTE: When SUPERVISOR is restarted after a HALT LOAD or QUIT or is started with '??RUN', and this option is not checked then SUPERVISOR will set AUTODC if necessary.

WAITOKTASK (Option Number 5)

The option WAITOKTASK increases the operator visibility of the `WAIT ("<TEXT>",OK)` statement used in an ODTSequence. As well as the usual message `'ENTER TT OK <TEXT>'` cycling each minute, SUPERVISOR will invoke a WFL job called `SUPERVISOR/RSVP/<ODTS>` which will enter the waiting entries with the following ACCEPT message:

```
ACCEPT: ENTER 'TT OK <ODTS>' <TEXT>
```

where `<ODTS>` is the name of the ODTSequence and `<TEXT>` refers to the text parameter in the WAIT statement.

If `'TT OK <ODTS>'` is entered, the ODTSequence will continue and the waiting entry will be automatically acknowledged.

NODEFACTIVE (Option Number 7)

If the NODEFACTIVE option is set, SUPERVISOR will Not allow the recompilation of an Opal routine which is currently in use, unless the `*` option is used in the define:

```
TT DEF * SITU MYSITU...
```

STRICTODTS (Option Number 8)

The **STRICTODTS** option enables the User to impose some control over the use of Opal String Expressions as Statements within the code of an ODTSequence. Opal treats isolated or 'dangling' Strings as ODT commands that are to be input to the System. While it is possible to code an ODTSequence using isolated Strings to generate the required System input message it is far better to avoid this practice and use the Opal ODT Statement to generate input messages. Isolated or 'dangling' Strings are discussed further in the **METALOGIC OPAL Programming Reference Manual**.

When an ODTSequence is DEFINEd either directly by terminal input or from an ENTER file the Opal compiler will now detect and report any String Expressions used as Statements as 'Unsafe constructs'. If the **STRICTODTS** option is **RESET** the compiled code will be stored in the SUPERVISOR Schedule but marked as 'Unsafe'. If the **STRICTODTS** option is **SET** the construct will be reported as a Syntax Error and no code will be stored.

When an ODTSequence is invoked either directly via a **TT DO** or as the target of a **TT WHEN DO** the 'Unsafe' marker is examined. If the code is marked as 'Unsafe' and the **STRICTODTS** option is **SET** the **DO** or the **WHEN** will not be started and an error message indicating that an attempt has been made to execute 'Unsafe' code will be displayed.

Further information on this option can be found in the [DEFINE](#) command and [DO](#) command sections of this manual.

MONITORING (Option Number 10)

This option logs all traffic between SUPERVISOR and the MCP CONTROLLER. Entries from OPAL programs appear under their sessions, but if they have no session, or if they are from scheduled AFTER inputs, they appear under the MONITOR session. Ordinary ODT inputs are logged as MCS message entries and WFL inputs are logged as RJE CC entries, but no JOB in Q, BOJ, EOJ, BOT, EOT or message entries will be logged since they are adequately logged by the system. The entry includes the time at which the activity was scheduled and the locking usercode (if it exists). Any responses from the system are also logged.

The MONITOR session number is returned in the response to the **TT USE** command, to enable LOGANALYZER to be run without releasing the session. SUPERVISOR displays a message like:

```
SUPERVISOR:MONITOR SESSION IS 5478
```

whenever a new MONITOR session is allocated, so the job summary of SUPERVISOR (which always appears) points to the session number.

The sessions are deallocated whenever a program terminates, or if a restart of SUPERVISOR occurs or a SUPERVISOR TL command is executed. The MONITOR session is split whenever SUPERVISOR restarts or a TL command is executed or if **MONITORING** is reset. In addition it will be split should an **so** command set **MONITORING**, even if it were already set. This behaviour allows printing the job summary of the MONITOR session without losing any logging information.

The log entries made under **MONITORING** are also visible in a LOG context OPAL or by using the SYSLOG command.

See Also

[HARDCOPY](#) Option

LOGOPERATOR (Option Number 13)

The LOGOPERATOR option does an LJ system command for each WFL job as it leaves the system queues and enters the mix. The text of the LJ command contains a list of the name of all operators currently known to SUPERVISOR. SUPERVISOR becomes aware of operators when they "log on" through the SHIFT BEGIN

command. If there are no operators known to SUPERVISOR, the LJ is still performed but the list of names is assigned the value "NONE".

See Also

[NAMEOPERATOR](#) Option
[SHIFT](#) Command

NAMEOPERATOR (Option Number 18)

The NAMEOPERATOR option insists on operator identification with a SHIFT BEGIN or SHIFT END command. SUPERVISOR validates operator-supplied identification by verifying that the identity is a valid ACCESSCODE associated with SUPERVISOR's own usercode. If the ACCESSCODE has an ACCESSPASSWORD, the password must also be entered.

See Also

[LOGOPERATORS](#) Option
[SHIFT](#) & [USE](#) Commands

USDATES (Option Number 19)

The convention for specifying dates in the United States is MM/DD/YYYY; and elsewhere it is generally DD/MM/YYYY. If USDATES is set then SUPERVISOR will require commands such as AFTER to use the MM/DD/YYYY convention otherwise DD/MM/YYYY.

Where applicable, SUPERVISOR's command responses will show dates in this format.

TAPELIB (Option Number 22)

When set, this option enables the TRIM Module of SUPERVISOR, which maintains the METALOGIC Tape Library Database. The database cannot be used without it. Setting TAPELIB is not permitted unless a valid key for TRIM is installed. Once TAPELIB has been set, SUPERVISOR will automatically invoke the TAPELIBUPDATER process to open the METATAPELIB database.

Please refer to the **Metalogic TRIM Reference** manual for more information

NOTTLOGGING(Option Number 28)

If this option is set then all command and response logging is suppressed from any TT function call.

DEFINERESP (Option Number 30)

This option enables the user to make the response to a **DEFINE** ? command a little safer, at the expense of needing to type an extra '+' when a modification is desired. With the option reset, **DEFINE** ? will return "DEFINE +" In the response, making it simple to modify the DEFINE content. If the DEFINERESP option is set, the response changes the heading line to **DEFINE** ?. This feature is provided to avoid inadvertent modification of DEFINES.

For example:

```
DEF ? ODTs TEST
```

returns

```
DEFINE + ODTSEQUENCE TEST (MX) : ODT (MIXNUMBER, "AX")
```

After entering TT SO + DEFINERESP, it returns

```
DEFINE ? ODTSEQUENCE TEST (MX) : ODT (MIXNUMBER, "AX")
```

ASYNCWFL (Option Number 31)

The **ASYNCWFL** option controls the way WFL START commands, previously scheduled by the AFTER command, are handled by SUPERVISOR's GRINDER process when the JOB file is subsequently missing.

If ASYNCWFL is reset, then SUPERVISOR will issue a warning message and automatically DS the waiting entry. However, if the option is set, GRINDER invokes the WFL job asynchronously and generates a waiting entry. It is then the responsibility of the operator to load the file or to manually DS the waiting entry.

NOWHENRESTART (Option Number 32)

Normally, when SUPERVISOR restarts, all OPAL programs previously active are restarted as well (if possible). If the option **NOWHENRESTART** is SET, SUPERVISOR will still examine the restart information for previously active OPALs and release any of their sessions, but will NOT restart them. This option should only be set when the site provides its own mechanisms to re-establish programs after a SUPERVISOR restart.

NOVALIDATE (Option Number 33)

During SUPERVISOR initialisation, all areas of the SCHEDULE file are validated which, for a file with many areas in use, can take a considerable time to complete. The **NOVALIDATE** option, if set, will bypass the checking of all OPAL programs in the dictionary. This option will improve SUPERVISOR initialisation time, but should be used with discretion.

In the event that a SCHEDULE file has become corrupted because of an invalid

program entry, this is usually enough to halt SUPERVISOR completely during validation, unless the NOVALIDATE option is set. However, if the option is reset, it is possible to manually force no validation, by setting the Magus configuration variable SUP_NOVALIDATE to TRUE, using the Metalogic INSTALL utility from CANDE:

U META/INSTALL CONFIG

And changing the SUP_NOVALIDATE setting by entering the Supervisor sub menu.

AUTOFS (Option Number 35)

If jobs or tasks were being scheduled by the system, for example because a system-wide HS command had been issued, any SUPERVISOR housekeeping tasks such as SAVE or PRINT commands and OPAL compile tasks would naturally also be scheduled.

However, if the **AUTOFS** option is set, SUPERVISOR will automatically FS (Force Schedule) any scheduled tasks or jobs originally invoked by SUPERVISOR itself.

NOSTATS (Option Number 36)

SUPERVISOR maintains internal statistics tracking CPU usage in various areas throughout the program visible with the SHOW UTIL command. Although this information is useful, there is an overhead associated with the maintenance of the data and, on metering systems in particular, this facility may now be controlled.

If the NoStats option is set then SUPERVISOR will discard all CPU statistics information and disable usage monitoring. The option may be turned on and off at any time as deemed necessary but some of the statistical percentage values in a SHOW UTIL response (plus the value for 'Unaccounted CPU time') will be incorrect. On busy systems, turning NOSTATS on may save significant CPU seconds over a long period of time.

NOENTERLIST (Option Number 37)

If set, the taskfile listing generated by an ENTER command will be inhibited. This allows sites to readily syntax check large ENTER files with the SYSTEM/OPAL compiler and thus prevent the generation of even larger print files from the OPAL compiles when the file is finally ENTERed.

This behaviour can also be controlled using dollar (\$) options in the ENTER file. If a listing is required with the NOENTERLIST option set, the inclusion of a text line consisting of '\$SET LIST' or just '\$LIST' will force a listing to be generated after that line.

Similarly, a '\$RESET LIST' will inhibit a taskfile print file if NOENTERLIST is not set.

Using multiple SET or RESET directives within the ENTER file will control the content of the generated print file.

By default, the NOENTERLIST option is RESET.

Note that \$SET LIST or \$RESET LIST will not be recognised by SUPERVISOR if they are used inside a DEFINE block.

AUTOTL (Option Number 38)

The AUTOTL option provides a mechanism for SUPERVISOR to track SUMLOG releases by Supervisor TL, ODT TL and MCP closures due to file size. After a TL, if AUTOTL is set, SUPERVISOR will COPY and REMOVE the closed SUMLOG to USE FAMILY FOR LOGS, if different from DL LOG, and also invoke any job specified by USE JOB AFTER TL.

If AUTOTL is reset, SUMLOG processing will only occur after a SUPERVISOR TL command, as in earlier implementations.

USECOPYWRITE (Option Number 45)

This option can only be set when there is a valid CopyWrite Key. When set, SUPERVISOR will use CopyWrite to access files that are at a Remote Host, instead of BNA.

COMPRESSDEFINE (Option Number 46)

If this option is set (SO+COMPRESSDEFINE) then SUPERVISOR will use Metalogic's MZIP technology to compress Opal DEFINE sources as they are written into the SCHEDULE file. Although there is a CPU overhead associated with deflation (DEFINE +) and inflation (DEFINE ? and SAVE DEFINE), most DEFINE sources can compress to 20-25% of the original size or better for larger Opals.

Since the SO COMPRESSDEFINE setting applies to all DEFINE + or ENTER actions, it is possible to override the default action by using the SET or RESET COMPRESS directive with an individual DEFINE:

```
DEFINE + ODTSEQUENCE TEST(MX) SET COMPRESS :  
DEFINE + ODTSEQUENCE TEST2(PER) RESET COMPRESS :
```

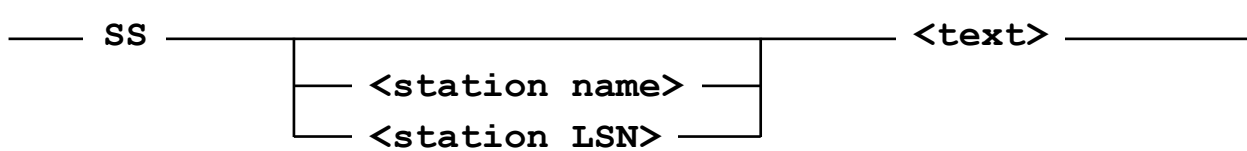
Note that SUPERVISOR does not compress Opal sources that are less than 2 segments (360 chars) in length.

With the recent increase in the size of OPAL scripts that can be stored in the SCHEDULE from 255 to 2184 segments (see SUPERVISOR DNote 541.74) then even larger scripts can be stored if they are compressed with MZIP.

LOGGING (Option Number 47)

The LOGGING option controls if SUPERVISOR will log all user activity, unsolicited messages and errors to his private log file, accessible with the [LOG](#) command. If LOGGING is RESET (default), the LOG command is inactive.

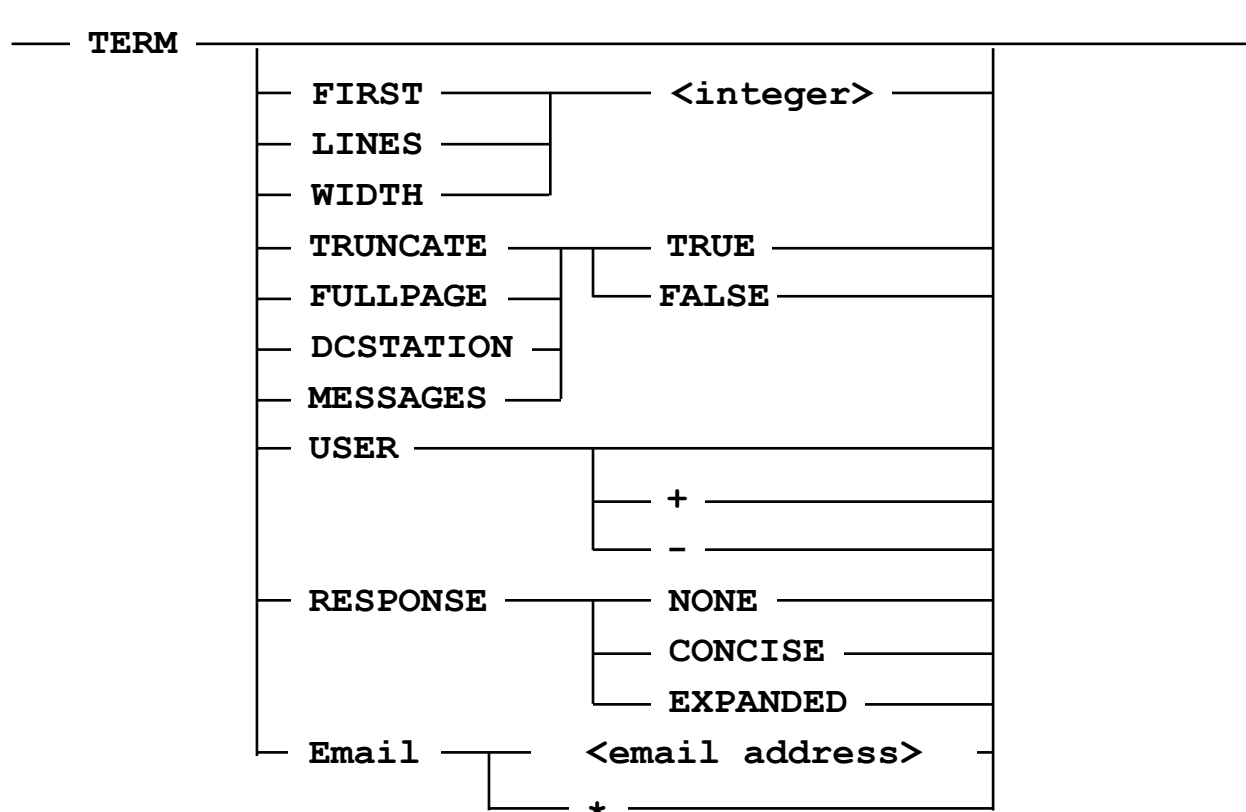
Send Station (SS) Command



The SS command is analogous with the same command in other MCSes. It allows text to be sent to either a designated Window station or to all Supervisor COMS window Stations that are open at that time.

If a specific station is to receive the message, the selected station may be designated by either its LSN or station name.

TERM Command



TERM is only accepted at a SUPERVISOR WINDOW or REMOTESPO and emulates the behaviour of the TERM system command. For details on the syntax and meaning of <ODT terminal specifications>, refer to the **Unisys System Commands Reference Manual**.

Although it has a similar syntax, the behaviour of the MESSAGES variant is slightly different in SUPERVISOR. When MESSAGES is TRUE, any SUPERVISOR error messages will be displayed. In addition, for Window Stations, system messages generated by requests handled by Controller e.g. CHANGE, COPY, REMOVE and MSG commands will be displayed. Messages generated by STARTJOBS etc will not be returned.

SUPERVISOR now supports TERM specifications up to 47 lines and 96 characters on SUPERVISOR COMS windows. Depending on the emulator, COMS will pass information about a stations TERM setting automatically to SUPERVISOR when the window is opened.

The USER variant allows the user to input or change a usercode associated with a terminal and also specify whether the usercode should be made 'active'. If a terminal has an associated usercode and it is made 'active', then all inputs to SUPERVISOR behave as if they had a corresponding FOR qualifier. The response to a DEF ? command will, however, not show a FOR header. Establishing a TERM USER specification can avoid the continual re-input of a password when interrogating or invoking locked DEFINES.

When a user opens a SUPERVISOR COMS window any usercode and accesscode associated with the station are passed to SUPERVISOR and stored along with any other terminal attributes made available by SYSTEM/COMS. However, SUPERVISOR will not associate this usercode with commands until it is made

'active' as described later in this section. A TT TERM command will return a response showing the usercode and accesscode, if any, together with other terminal attributes. The response will also show any PRINTDEFAULTS associated with the usercode as illustrated below:

```
TERM
TT TERM LINES 24 WIDTH 80 FIRST 2 TRUNCATE TRUE RESPONSE EXPANDED
      MESSAGES TRUE FULLPAGE FALSE DCSTATION TRUE
      You are: BOBSMALL/S/1 [175]
      Log-On User: BOB
      PrintDefaults: destination="MAIL",Note="To:bob.1.MAIL@metalog.com"
      Email: BOB.1.MAIL@METALOG.COM
```

Any printed reports generated from this station, e.g. commands prefixed by the PRINT modifier or OPAL programs that write to print files, will inherit the PRINTDEFAULTS from the original COMS usercode, if applicable. If PRINTDEFAULTS is not assigned to the usercode, then SUPERVISOR will use the default USE DESTINATION setting.

Terminal attributes may be changed in the usual way, for example:

```
TERM FIRST 2
TERM MESSAGES NO
```

and will be confirmed by an updated TT TERM response.

A usercode can be made active on a window station by entering a TT TERM USER + command. This will return a response to confirm that the usercode has been activated, for example:

```
TERM USER +
TT TERM LINES 24 WIDTH 80 FIRST 2 TRUNCATE TRUE RESPONSE EXPANDED
      MESSAGES TRUE FULLPAGE FALSE USER BOB DCSTATION TRUE
      You are: BOBSMALL/S/1 [175]
      Log-On User: BOB (In use)
      PrintDefaults: destination="MAIL",Note="To:bob.1.MAIL@metalog.com"
      Email: BOB.1.MAIL@METALOG.COM

      Note: Opal programs entered using the DEF command will be stored
            with this Usercode attached.
```

Similarly, an in-use usercode can be de-activated by entering a TERM USER - command, returning a response to confirm that the usercode is no longer in use. The current PRINTDEFAULTS setting will remain permanently assigned to the station, irrespective of the use of TERM USER + or TERM USER - commands.

To change a usercode associated with a terminal the unqualified form of the TT TERM USER command is used. A 'formed' screen, as shown below, will then be displayed to allow entry of the Usercode and any associated Password. The password, if required by the Usercode, will be entered on the screen in 'secure video' (if the emulator supports this facility) verified and then discarded - it is not stored within SUPERVISOR. Transmitting blank fields will terminate the usercode entry dialogue and leave the existing usercode unchanged.

Please enter your Usercode

... and your Password

(Transmit blank fields to abort entry)

After the usercode and any associated password has been verified a further screen will be displayed for entry of an accesscode if this is required for the usercode.

Entry of blank fields will cause SUPERVISOR to prompt for re-entry of the usercode. After the accesscode has been entered and verified, the **TT TERM** response screen will again be displayed to confirm that the changes have been stored. This screen will now indicate that the usercode has been changed, by displaying the legend 'Current User', (a usercode passed by COMS is denoted by 'Log-On') and 'In Use' is not displayed indicating that the new usercode is not currently active. A **TT TERM USER +** is required to activate it.

Note:

The FOR modifier may be used for the 'Current' or 'Log On' with no password.

Ex.

If the current user FLEX with Accesscode RULES then

FOR FLEX DO FLEX_DUMP

would run the ODTs FLEX_DUMP with User Flex Access Rules

For example:

```
TT TERM LINES 24 WIDTH 80 FIRST 2 TRUNCATE TRUE RESPONSE EXPANDED
MESSAGES TRUE FULLPAGE FALSE DCSTATION TRUE
You are: BOBSMALL/S/1 [212]
Current User: FLEX Accesscode: RULES
Email: BobAndIan@metalog.com
```

From SUPERVISOR windows only, it is possible to override the USERDATA-assigned email default normally retrieved by SUPERVISOR from the usercode PRINTDEFAULTS or EMAIL attributes.

```
TERM EMAIL support@metalogic.eu.com
TERM EMAIL *
```

In the first example, the provided email address temporarily overrides the usual default usercode email. The assignment remains in force until the window is closed by the user or a SUPERVISOR restart. No validation is performed on the email

address and it must be less than 60 characters in length. Both the usual and override email addresses are shown in a TERM response.

The temporary email address can be removed by using the TERM EMAIL * command.

See Also:

[FOR Modifier](#)

TapeLibUpdater (TP) Command

TP

<text>

The TP command passes the supplied text to the TAPELIBUPDATER program together with sufficient information for TAPELIBUPDATER to identify the origin of the message and the usercode associated with the origin (if any). SUPERVISOR does not perform any parsing of <text>.

For further information on TAPELIBUPDATER, refer to the **Metalogic TRIM Manual**.
The TP command will be rejected if the SUPERVISOR TAPELIB option is not set.

USE Command

USE

<BDNAME spec>

<CHARGE spec>

<DESTINATION spec>

<FAMILY spec>

<FILE spec>

<GETSCRATCH spec>

<JOB spec>

LANGUAGE <integer>

<MODE spec>

ODT <integer>

QUEUE <integer>

<SCRATCHPOOL spec>

<SILO spec>

<TASK spec>

<USER spec>

NOPASSWORDS

+ -

<WINDOWSECURITY spec>

USE may not be the subject of an AFTER command.

USE USERCODE and USE TAPELIB USER will only be accepted if preceded by a FOR Modifier specifying a privileged usercode, or from a system console with a TERM USER set to a privileged usercode.

USE always returns a response similar to the following display, though if an item is not assigned, it may not appear:

```
USE
      ---- USER and FAMILY settings ----
USER      : SUPERVISOR      USER for TAPELIB      : TAPELIB
USER for ODTSECURITY : META      USER for EXTERNAL     : META
USER for MAIL        : META      FAMILY for NAPLOGS      : VMSA
FAMILY for EXTERNAL  : DEV        FAMILY for SAVES       : DEV
FAMILY for LOGS      : DEV        FAMILY for SYSTEM      : DISK
FAMILY for SCHEDULE  : DISK
      ---- TASK and FILE settings ----
FILE for REBUILD     : (META)SYMBOL/SCHEDULE ON DEV
JOB after TL         : (IPP)TL/HANDLER ON DEV
TASK for RECORDER    : Not specified
TASK for NAPSTATSLIB : *NAPSYSTEM/NSL ON VMSA
FILE for NAPRUNLIGHT : *NAP/RUNNING/LIGHT ON VMSA
      ---- Miscellaneous settings ----
QUEUE              : 255      BACKTRACK mode      : AUTO
ODT                : 1        LANGUAGE            : ENGLISH
WINDOWSECURITY     : MAXIMUM   PASSWORD VALIDATION : NOT ACTIVE
DESTINATION        : MAIL:bobandian
SILO SUPPORT       : UNISYS
```

All of the USE settings are held as MAGUS configuration variables instead of in the SCHEDULE file and are preserved in the symbolic file generated by a SAVE SCHEDULE AS <file> command. Some commands allow the NONE or '-' modifier to cancel the current USE setting.

The following describes each USE modifier in alphabetical order:

USE BDNNAME

<BDNAME spec>

— BDNNAME ———— <identifier> ————
 └── _ ───────────┘

The BDNNAME option forces all print files created by SUPERVISOR to be held in a user-specified directory instead of printed by the Print System.

If set, printfiles will be under the SUPERVISOR usercode and a directory of <BDNAME>/<YYYYMMDD>

Example

```
(SUPERVISOR) SUPBD/20080305
```

The BDNNAME assignment can be cancelled using the '-' modifier.

USE CHARGE

<CHARGE spec>



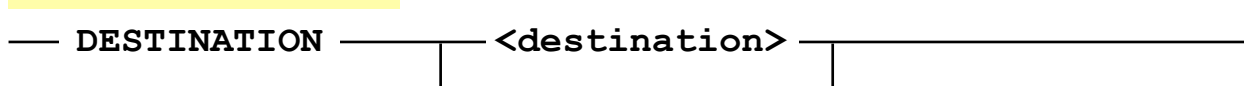
The **CHARGE** variant of the **USE** command sets a default **CHARGECODE** to be assigned to many **SUPERVISOR** command processes and housekeeping tasks.

The **CHARGE** assignment can be cancelled using **USE CHARGE -**

If no **CHARGE** is assigned, all tasks will run with a null value unless **CHARGEREQ** is set on any usercode assigned to the process. If assigned, **CHARGE** is only conferred if **CHARGEREQ** is set on the task usercode.

USE DESTINATION

<DESTINATION spec>



The **DESTINATION** variant of the **USE** command sets the default printer destination to be used for all print files generated by **SUPERVISOR** or its subtasks.

The ' - ' (minus) modifier allows the current **DESTINATION** specification to be removed.

USE DESTINATION meta

USE DESTINATION -

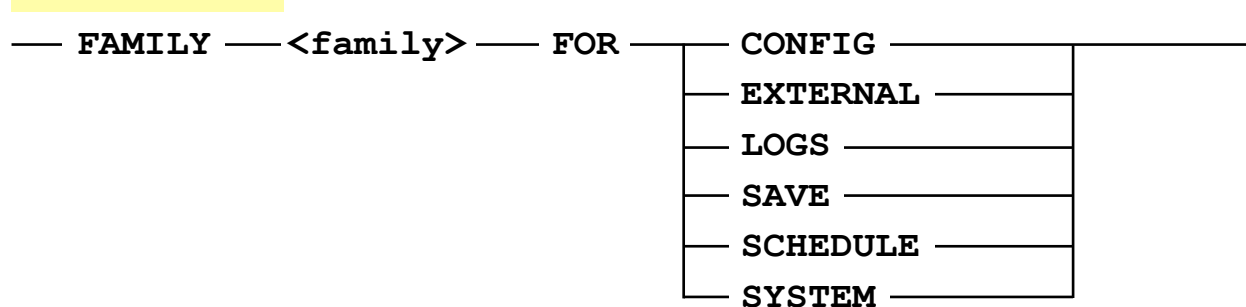
However, for printable **SUPERVISOR** commands generated from any **COMS** window station that has an associated Usercode, **SUPERVISOR** will now automatically assign any **PRINTDEFAULTS** already associated with the usercode in the **USERDATAFILE**. Note that the Usercode does not NOT have to have been activated (**TERM USER +** is not needed).

This change means that **SUPERVISOR** commands, such as **ENTER** or any **TT PRINT** variant, issued from a **COMS** window will generate print requests with printing defaults inherited from the caller.

The response to a **TT TERM** command will show the **PRINTDEFAULTS** if any have been established.

USE FAMILY

<FAMILY spec>



FOR CONFIG

This special **USE** variant should only be used where more than one Metalogic MAGUS "configuration" needs to be supported on the same system. Typically, such a situation arises for Unisys 1+1 systems where a standby system, running a minimum disk configuration, may be switched to replace a failed live system, which has a large disk subsystem.

When a **USE FAMILY... FOR CONFIG** is initiated, SUPERVISOR verifies that an alternative run-time MAGUS configuration exists on the specified family. The file is called:

***METALOGIC/MAGUS/CONFIGURATIONDATA**

If a valid file is resident, SUPERVISOR initiates the closedown of all SUPERVISOR WHENS, closes the SCHEDULE file and calls the MAGUS library to switch control to the new configuration file.

This allows the customer to maintain two separate SUPERVISOR environments; in particular, the **USE FAMILY... FOR SCHEDULE** setting allows two completely different SCHEDULE files to be maintained (different OPAL programs, SUPERVISOR options, active WHENS, AFTERS etc.).

USE FAMILY VMSA FOR CONFIG

USE FAMILY - FOR CONFIG

Using the '-' modifier cause SUPERVISOR to revert to the standard MAGUS CONFIGURATIONDATA file, which will be resident on the family used by the SL-ed MAGUS.

If SUPERVISOR detects a change in the current CONFIG setting, the SCHEDULE file is closed, all active WHENS are terminated and a "pseudo"-restart is performed. The MAGUS library loads the specified CONFIGURATIONDATA file and SUPERVISOR retrieves all run-time information applicable to the new environment. The SCHEDULE file, as specified in the new configuration, is then re-loaded and all previously active WHENS are restarted.

The CONFIG modifier is the only variant of USE FAMILY that is allowed to use this feature.

More information on this feature can be found on the Metalogic web site at <http://www.metalogic.eu.com/Main/docum/Techinfo/switch.pdf>

FOR LOGS

USE FAMILY... FOR LOGS designates the family where SUMLOGs should be copied to, during the TL command.

USE FAMILY DEV FOR LOGS

If this nominated family and the DL LOG family have the same name, then SUPERVISOR will NOT attempt to copy any SUMLOG files found after the **TL** command has been issued.

If the nominated family and the DL LOG family are different and the Supervisor user code does not have the Secadmin privilege set, the command will return an error.

FOR NAPLOGS

USE FAMILY... FOR NAPLOGS designates the family where SUPERVISOR can find NAPLOG files generated by Unisys NAP system software. The setting is used by the SHOW NAPLOGS command as the primary disk family that should be searched.

USE FAMILY VMS FOR NAPLOGS

See also

[SHOW command](#)

FOR SAVES

USE FAMILY FOR SAVES sets the default destination of the SAVE and the default source for RELOAD commands. The default value (the same family as JOBDESC, which is set by the "DL JOBS" system command) is bound into SUPERVISOR's code file to facilitate recovery after a COLD START or the loss of the SCHEDULE file.

USE FAMILY DEV FOR SAVES

FOR SCHEDULE

By default, SUPERVISOR expects to find its *SCHEDULE file on the family assigned to DL JOBS. However, the user can override this configuration by using the USE FAMILY...FOR SCHEDULE command; if this command is issued, then SUPERVISOR will close the *SCHEDULE file and COPY it to the new family. This family will then be searched first on subsequent SUPERVISOR restarts.

USE FAMILY WORK FOR SCHEDULE

If, for any reason, the *SCHEDULE file cannot be moved, for example a *SCHEDULE file is already present on the destination family, or the COPY fails for some reason, the move will be rejected with:

SCHEDULE COULD NOT BE MOVED

and no change will be made.

FOR SYSTEM

USE FAMILY... SYSTEM alters the name of the disk pack families where SUPERVISOR expects to find system code files (for the SYSDIR, JAM, PB, TDIR commands plus the SYSTEM/HARDCOPY utility).

USE FAMILY DISK FOR SYSTEM

When SUPERVISOR's default queue (see USE QUEUE) is created, the family equation assigned to the queue will use this family as primary. The queue is only ever modified during SUPERVISOR initialisation.

FOR EXTERNAL

USE FAMILY ... EXTERNAL assigns a default FAMILY for the commands SAVE AS, ENTER FROM, or DEFINE FROM if an " ON " part is not provided with the filename.

USE FAMILY DEVELOPMENT FOR EXTERNAL

USE FILE

<FILE spec>

— FILE — <filename> - FOR — REBUILD —
 NAPRUNLIGHT

FOR NAPRUNLIGHT

This variant is ONLY applicable to sites that are using Unisys NAP (Network Application Platform) hardware and software. The designation of this file title informs SUPERVISOR where to find the NAP RUNNING/LIGHT file, whose open state is checked by SUPERVISOR to ensure that NAP is active and accessible.

USE FILE *NAP/RUNNING/LIGHT ON VMSA FOR NAPRUNLIGHT

FOR REBUILD

The USE FILE... REBUILD variant allows sites to specify the filename of the default OPALS file to be used during a SUPERVISOR restart when the *SCHEDULE or suitable *SAVED/SCHEDULE files cannot be found.

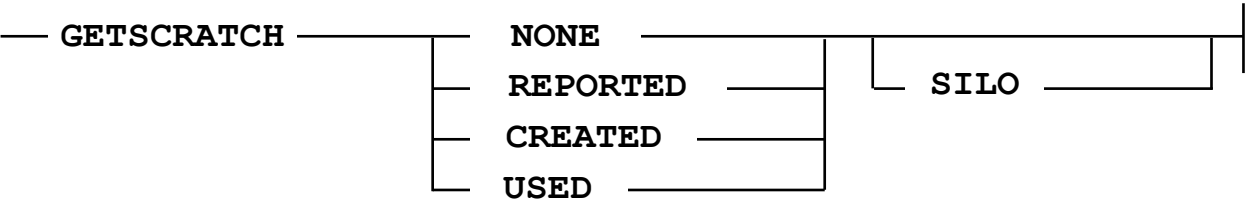
USE FILE (META)DECOMPILED/SCHEDULE ON DEV FOR REBUILD

If the file specified in the USE FILE <filename> FOR REBUILD does not explicitly specify usercode or family then the 'User for External' or 'Family for External' will be used, respectively.

If no USE FILE FOR REBUILD file is named, SUPERVISOR will search for a file called:

```
(<SUP usercode>) SAVED/SCHEDULE/DEFAULTS ON <Family for SAVES>
```

USE GETSCRATCH



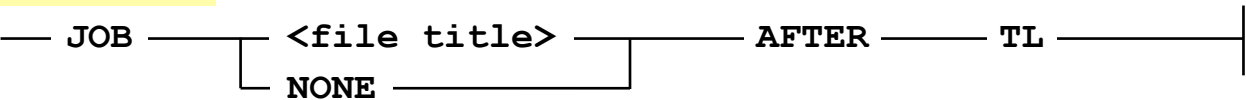
This variant is only valid on Trim systems and will only have an effect on Systems using the Metalogic Tapemanager. If set to a value other than NONE then TAPEMANAGER will attempt to assign scratch tapes to tasks requesting output tapes. If SILO is set then only tapes in the tape silo will be assigned. If no tapes are available in the silo the serial 'SCNONE' will be assigned. If SILO is not specified then if a Silo is available and contains scratchable tapes then a tape will be assigned from the silo, otherwise any scratchable tape will be assigned. The main parameter determines how scratchable tapes are selected. REPORTED means select the tape with the oldest SendDate. CREATED means select the tape with the oldest creation date. USED means select the tape with the lowest use count.

The Getscratch call in Tapemanager may be suppressed for selected scratchpools even if TT USE GETSCRATCH is set. This is controlled by the config variable TL_EXPOOLS which should be set to a list of pools to be excluded from Getscratch calls.

```
Ex TT / ($TL_EXPOOLS.Config:="NONSILO,"AUTOLOADER")
```

USE JOB

<JOB spec>



USE JOB ... AFTER TL selects the name of the job symbolic file after a SUPERVISOR TL command has been executed. Note that the job will be invoked for each SUMLOG found on the DL LOG family, not just the recently released SUMLOG file. For example:

```
USE JOB (META)LOG/ANALYSIS ON DEV AFTER TL
```


The job nominated as the **JOB AFTER TL** must have two parameters as follows.

```
BEGIN JOB ANALYSE/LOG (STRING LOGTITLE, INTEGER FUNCTION) ;
```

When SUPERVISOR starts the job, LOGTITLE will contain a file title with an "ON <family name>" part. The value of FUNCTION is the integer value supplied in the TL command or zero if no <integer> was specified. A suitable example is supplied on the release media in the file:

```
METALOGIC/JOB/ANALYSE/LOG
```

This is the default name set during SUPERVISOR's installation.

Alternatively, the automatic invocation of a default WFL job to process each log after the **TL** can be cancelled by:

```
USE JOB NONE FOR TL
```

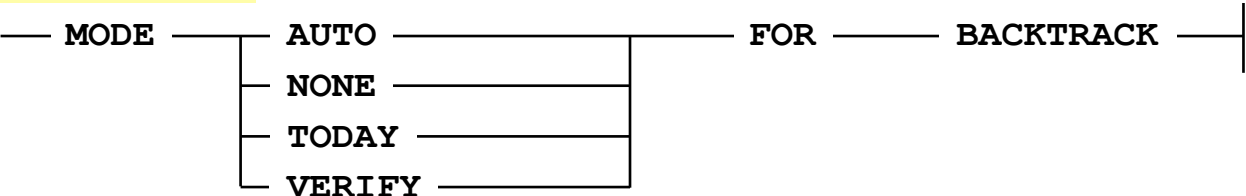
USE LANGUAGE

USE LANGUAGE changes the language of the messages generated by SUPERVISOR. The languages available at present are:

- 0 ENGLISH 1 DEUTSCH
- 2 AFRIKAANS 3 FRANÇAIS

USE MODE

```
<MODE spec>
```



The **USE MODE ... FOR BACKTRACK** has replaced the **AUTOBACKTRACK** option implemented in older releases of SUPERVISOR. The setting comes into effect after a **TT HS-** command has been issued to resume a held schedule, after a SUPERVISOR restart or when SUPERVISOR has detected that missed **AFTER** activities have not been processed for some reason. The **TT HS** command is described in detail earlier in this chapter.

Once the schedule has been resumed, SUPERVISOR invokes a special task called **BACKTRACK/SCHEDULE** to handle the rollback of the schedule, to the previous day if necessary, to begin processing any missed activities.

At this point, the manner of the back tracking operation depends on the setting of the **USE MODE ... FOR BACKTRACK**. The backtrack options currently available are:

```
TT USE MODE AUTO FOR BACKTRACK
TT USE MODE NONE FOR BACKTRACK
TT USE MODE TODAY FOR BACKTRACK
TT USE MODE VERIFY FOR BACKTRACK
```

If BACKTRACK mode is AUTO, then SUPERVISOR will automatically re-process all missed activities, starting with the remainder of the schedule of the day preserved at the time of the initial HS+. This is then followed by a reload of the current days schedule, if required. The schedules for any intervening days will NOT be processed; this applies regardless of the setting of BACKTRACK.

If BACKTRACK is NONE, then SUPERVISOR will automatically discard all missed activities from all schedules. TT AF? will show these discarded activities marked as "skipped".

If BACKTRACK mode is TODAY then SUPERVISOR will only process the missed actions from the current day's schedule; any earlier schedule will be loaded and processed but any missed activities will be skipped.

If BACKTRACK is VERIFY, then SUPERVISOR requires operator intervention to confirm the processing of each missed activity. The interface provided works most effectively from an ODT but interaction is also possible from a SUPERVISOR COMS window. Each command requires a TT SKIP or TT OK to, respectively, reject or allow the activity to be processed.

From each ODT, during a VERIFY backtrack, the following screens may be seen:

```
----- BACKTRACK SCHEDULE for THURSDAY 25/11/99 -----

BACKTRACK mode is currently 'VERIFY' and requires operator intervention

There have been 3 missed activities since 14:22

You may enter:
  'TT OK'           to see all missed activities
  'TT SKIP'         to skip all missed activities

Alternatively, you may change BACKTRACK MODE using 'TT USE'
```

If TT OK is entered, BACKTRACK/SCHEDULE will require verification of each remaining activity in the current schedule.

In the screen below, various options are available to the operator:

```
----- BACKTRACK SCHEDULE for THURSDAY 25/11/99 -----  
  
BACKTRACK has processed 0 of 1 missed time slots (0 skipped)  
BACKTRACK requires operator intervention for the following activities  
  
----- SCHEDULE after 17:13 TODAY -----  
1: DISPLAY("HI")  
  
BACKTRACKSCHEDULE will automatically discard any remaining activities in  
180 seconds unless intervention has occurred  
  
You may enter:  
  'TT OK'           to process this missed activity  
  'TT OK + nn'      to re-schedule in 'nn' mins  
  'TT SKIP'         to omit this missed activity  
  'TT SKIP *'       to omit ALL remaining activities  
  'TT OK *'         to process ALL remaining activities  
  
Alternatively, you may change BACKTRACK MODE using 'TT USE'
```

At this stage, entering TT OK causes SUPERVISOR to immediately process the activity. Using the TT OK * variant causes all missed activities to be processed immediately, as if USE MODE .. BACKTRACK was set to AUTO.

Alternatively, SUPERVISOR can re-schedule the activity for some time in the future e.g. TT OK + 60 requests SUPERVISOR to reassign the activity to be performed in one hours time. The minimum time is 30 minutes; the latest is a calculated scheduled time up to 23:59. If the backtrack is taking place on a schedule that is not current ie. a previous day, the re-time will be rejected.

TT SKIP causes the activity to be cancelled; its postscript will be marked as "skipped". The variant TT SKIP * causes all missed activities to be skipped with no further operator intervention.

This behaviour can be changed at any stage by altering the setting of USE MODE. Within 15 seconds of changing the MODE, SUPERVISOR will handle the remaining activities depending on the new USE MODE setting.

By default, all currently active SUPERVISOR COMS windows will receive a notification on the Status line when a backtrack process starts and terminates.

During a schedule resumption where BACKTRACK mode is VERIFY and a user on a Supervisor COMS window has been made aware that a backtrack is running, it is possible to divert operator control to that station by entering TT OK. All subsequent acknowledgement screens will be sent to this station as well as the normal system ODTs.

USE ODT

USE ODT tells SUPERVISOR which ODT should, by default, receive SUPERVISOR-related system messages program. The value of <integer> may not

exceed 255. If the ODT specified is not usable then SUPERVISOR will use the available system console with the lowest unit number.

USE QUEUE

USE QUEUE sets the queue or class for all jobs started by SUPERVISOR. The value of **<integer>** may not exceed 255. SUPERVISOR will create or modify the attributes of this queue each time it is initiated. If the queue does not exist at SUPERVISOR's initialisation time, it will be created as:

```
MQ <integer> MIXLIMIT 1
      FAMILY DISK=<HL_PK> OTHERWISE <USE_SYS_PK>
```

SUPERVISOR only ever uses this queue for running the WFL job specified by **USE JOB ... AFTER TL** to handle SUMLOGs released by the TL command.

USE SCRATCHPOOL

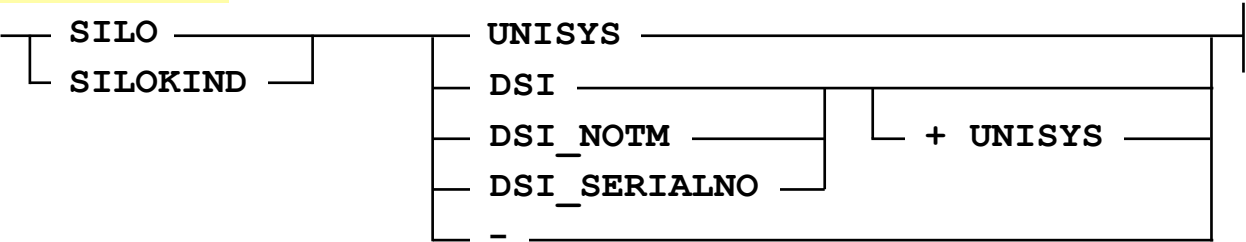
<SCRATCHPOOL spec>



USE SCRATCHPOOL, allows one or more tape scratch pool identifiers to be maintained in a list and identifies tapes that belong to the DSI robot.

USE SILO

<SILO spec>



The **USE SILO** command allows the TRIM Tape Library system to control and retrieve information from DSI or Unisys StorageTek tape robot silos. The modifier **<silo type>** controls the type of robot subsystem and the SUPERVISOR interface:

```
TT USE SILO DSI
TT USE SILO DSI_NOTM
TT USE SILO DSI_SERIAL
TT USE SILO UNISYS
USE SILO DSI + UNISYS
```

DSI means that full DSI Cartridge Tape Library support is enabled. All tape

assignments are passed through to the DSI CSCLIB by Metalogic's TAPEMANAGER for processing by DSI's own TAPEMANAGER library.

The DSI TM command is available from Supervisor windows.

HELP TM

----- SUPERVISOR HELP -----

TM command:

The TM command is only valid if the Metalogic TapeManager is SLed. It is used to set options controlling the Metalogic Tapemanager and optionally to pass commands to DSI tape library software.

```

TM  +-----+-----+-----+-----+-----+-----+-----+-----+-----+
    +- GETSCRATCH +- NONE -----+-----+-----+-----+-----+
    |              +- REPORTED -----+-----+-----+-----+-----+
    |              +- CREATED -----+-----+-----+-----+-----+
    |              +- USED -----+-----+-----+-----+-----+
    |              +- SILO -----+-----+-----+-----+-----+
    |              |               +-----+-----+-----+-----+
    |              |               |               |               |
    |              +- Exclude +- <ScratchPool list> ---+-----+
    |              |               +-----+-----+-----+-----+
    |              |               |               |               |
    |              +- NODENSITY + <ScratchPool list> ----+-----+
    |              |               +-----+-----+-----+-----+
    |              |               |               |               |
    +- DSIPPOOL +- <ScratchPool list> ---+-----+-----+-----+
    |              +-----+-----+-----+-----+
    +- LOCALTM +- <SL ID> ---+-----+-----+-----+-----+
    |              +-----+-----+-----+-----+
    +- DSISTART -----+-----+-----+-----+-----+
    +- DSIQUIT -----+-----+-----+-----+-----+
    +- <DSI Command> -----+-----+-----+-----+-----+

```

A TM command on its own will return all of the current TapeManager settings.

GETSCRATCH options

NONE Turns off the assignment of scratch or scratchable tapes for tape assinnment requests.

The next three options determine how scratchable tapes are selected.

REPORTED means select the tape with the oldest SendDate.

CREATED means select the tape with the oldest creation date.

USED means select the tape with the lowest use count.

SILO means only tapes currently in a tape library will be selected. SILO- removes the restriction.

EXCLUDE is used to specify a list of scratchpools which are excluded from GETSCRATCH handling. EXCLUDE - deletes the list.

NODENSITY is used to control whether Density will be used in selection of Scratchable tapes. Any request with a ScratchPool which appears in the specified list will

Window S21 at DELL MCP

```

TT +
----- SUPERVISOR HELP -----
TM command:      not have density used in tape selection. NODENSITY -
                  deletes the list.

DSIPOOL is used in mixed DSI/Unisys environments to identify the
ScratchPools allocated to DSI libraries. DSIPOOL - deletes
the list.

LOCALTM is used to specify the function name of an Sled library.
If specified Tapemanager will call the entryptpoint
CHECK_TAPE_ASSIGNMENT before performing its own
CHECK_TAPE_ASSIGNMENT to allow local settings such as
ScratchPool and Density. LOCALTM - Turns off the option.

DSISTART is used to start the DSI libraries, normally only
required after a DSIQUIT.

DSIQUIT is used to Quit the DSI libraries without bringing down
the TapeManager.

<DSI Command> may be any valid DSI command. Only valid if DSI has
been selected in the USE SILO command.

```

In order to detour a problem when calling DSI Commander entry point where TM CONFIG HOST messages seemed to be ignored an new config option has been added to allow an alternative to DSI CSCLIB to be called

```
TT ($TL_DSISUPPORT.Config:="DSISUPPORT")
```

to change from CSCLIB to DSISUPPORT

```
TT ($TL_DSISUPPORT.Config:=Empty)
```

To return to CSCLIB

The Config variable TL_DEBUG may be used to add debug information to the tape log. Turn on debug with

```
TT / ($TL_DEBUG.CONFIG:="TRUE")
```

Turn off debug with

```
TT / ($TL_DEBUG.CONFIG:=Empty)
```

Test debug with

```
TT / ("",$TL_DEBUG.CONFIG,"")
```

DSI_NOTM is similar to DSI except that DSI TAPEMANAGER runs in ACTIVE mode but any tape requests are NOT passed through to the DSI software by Metalogic's TAPEMANAGER.

DSI_SERIALNO is the same as DSI except that tape assignment requests are only passed to CSCLIB if a tape serial number has already been assigned to the request. Note that if both a SERIALNO and SCRATCHPOOL are assigned, the SCRATCHPOOL assignment is removed by the Metalogic TAPEMANAGER library.

UNISYS enables support for Unisys CSCLIB 2.1 and later for larger Tape Robot CTL systems such as StorageTek. The TK command is available for silo

interrogation. See Help TK for details.

Adding + UNISYS to any of the DSI variants indicates a mixed environment. Tape requests to use the DSI robot are identified by the specification of job SCRATCHPOOLS whereas all other robot requests will be routed to the Unisys hardware, if applicable.

The '-' modifier will immediately cancel the current USE SILO setting e.g.

TT USE SILO -

Changes to the USE SILO setting are applied immediately. As well as SUPERVISOR, both the TRIM software and Metalogic's TAPEMANAGER library are informed.

It is now possible to have Metalogic/Tapemanager call the entrypoint Check_Tape_Assignment in a local copy of Tapemanager before calling the Metalogic Check_Tape_Assignment. This is controlled by a config variable TL_LocalTM. If this variable is set to the SLED name of a library then we will call the entrypoint in that library first. The variable can be easily set from Supervisor.

Ex TT / (\$TL_LOCALTM.Config:="MYMANAGER")

USE TASK

<TASK spec>

— TASK — <file title> — FOR —
 └── RECORDER ───┐
 └── NAPSTATSLIB ───┘

FOR NAPSTATSLIB

This variant is ONLY applicable to sites that are using Unisys NAP (Network Application Platform) hardware and software. The designation of this file title informs SUPERVISOR where to find the Unisys NAP Statistics Library (NSLLIB), which is not SL-ed.

USE TASK *NAPSYSTEM/NSL ON VMS FOR NAPSTATSLIB

This library is used by SUPERVISOR to support both the NAPLOG OPAL context and NAP system attributes.

FOR RECORDER

The TASK .. RECORDER variant of the USE command sets the name of the codefile to be used as the external Metalogic RECORDER program. Metalogic provide both HOTLINE and RECORDER programs as standard but sites are at liberty to change them to suit their own requirements.

The default Recorder interface is via Library entry points. Setting USE TASK ...FOR RECORDER to anything other than *METALOGIC/SUPERVISOR/RECORDER will use the old (non library)interface.

It is strongly recommended that the default Metalogic version of recorder is used.

USE TASK *OBJECT/SITE/RECORDER FOR RECORDER

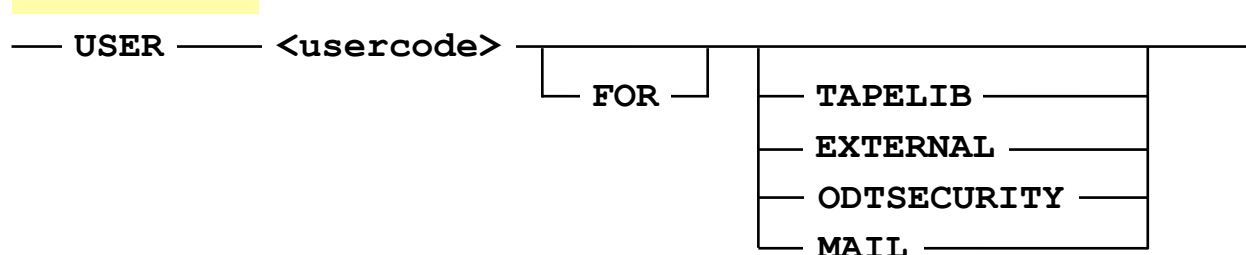
If USE TASK is not specified then the default RECORDER codefile will be named:

***METALOGIC/SUPERVISOR/RECORDER**

Please refer to [Hotline and Recorder](#) for more details on the RECORDER program.)

USE USER

<USER spec>



USE USER changes the private privileged usercode maintained by SUPERVISOR for its own purposes. This must be preceded by a FOR Modifier with a privileged usercode. See the FOR Modifier for details:

USE MYPW/PW USE USER SUPERVISOR

This usercode is used to group together, for charging purposes, all resources used by SUPERVISOR: all WHEN, DO, and EVAL commands allocate a session under this usercode. All task initiations (for example, SYSTEM/FILEDATA when a SYSDIR command is processed or SYSTEM/LOGANALYZER when a SYSLOG command is processed) are run under this usercode. All internal processes of SUPERVISOR are also initiated under this usercode.

To guarantee that security is preserved, the password of this usercode is changed every four hours. SUPERVISOR will usually unconditionally modify its usercode every 4 hours, including the update of its password.

Setting the Magus configuration variable called SUP_PWTIMEOUT may change this 4-hour interval. This can be changed using the Metalogic INSTALL utility

U META/INSTALL CONFIG

And change SUP_PWTIMEOUT via the Supervisor sub menu.

INSTALL allows changes for SUP_PWTIMEOUT to a maximum of 120 minutes; a SUPERVISOR restart is required to enforce the new setting. SUPERVISOR will ensure that the time-out period is always greater than 30 minutes and less than 2880 minutes, or 2 days.

SUPERVISOR will ensure that the selected usercode is privileged and the USERDATA attributes ANYOTHERCLASSOK is set. The CLASS attribute is set to the number specified in the **QUEUE** variant of the **USE** command; SYSTEMUSER is also set. If that usercode has a valid CANDEDESTNAME, it will be used for all tasks SUPERVISOR runs. If none is present, the default is SITE. If the usercode has a valid PRIORITY locator, its setting will be used on all tasks otherwise a default priority of 50 is used.

Entry of SUPERVISOR's USE USER <usercode> syntax will cause the old usercode to be deleted from the system USERDATAFILE. All accesscodes in the ACCESSCODELIST are also lost. If this occurs, SYSTEM/MAKEUSER must be run to create a new list of valid operator identifications.

If the NAMEOPERATORS option is set, SUPERVISOR looks at the standard

USERDATA attribute ACCESSCODELIST to decide if a given operator identification is valid or not.

FOR EXTERNAL

This variant of USE USER specifies the default usercode used for external files. (i.e. files used in SAVE AS, ENTER FROM or DEFINE FROM).

USE USER META FOR EXTERNAL

USE USER.. FOR EXTERNAL, in conjunction with USE FAMILY... FOR EXTERNAL, enhances commands such as

```
SAVE DEF META_ = AS META/FILES
```

with a default destination family and usercode.

FOR MAIL

If the Metalogic MAIL Library is installed, SUPERVISOR is able to send SMTP mail using the OPAL MAIL facility. By default, all SUPERVISOR generated mail will have a sender of the SUPERVISOR usercode. This can be changed to any other selected usercode by using the USE ... FOR MAIL variant, though as with other USE specifications, the command must be prefixed by a FOR modifier:

```
FOR METAPU/PW USE USER SMTP FOR MAIL
```

FOR ODTSECURITY

This variant of USE USER is assigned extra privilege. The nominated usercode may modify OPAL DEFINES, WHENs or scheduled activities locked by another usercode by prefixing the command with:

```
FOR <odtsecurity user>/<password> FOR <usercode>  
FOR ODTSEC/ODTPW FOR META
```

If a system is running with the security option 'SECADMIN Authorized' then Supervisor requires a SECADMIN usercode to set the USER for ODTSECURITY.

This capability of the ODTSECURITY usercode has an important use with the ENTER command. When an ENTER is done under this usercode, any FOR modifiers within the file do not need to have passwords. If no ODTSECURITY usercode is set, an ENTER from an ODT can also use FOR modifiers without passwords.

This usercode can also be used in conjunction with the SUPERVISOR RESTRICT command, which requires the ODTCONTROL module to be licensed from Metalogic. Any command input from MARC or programs running under the ODTSECURITY usercode, plus any SUPERVISOR COMS windows with a TERM USER set to this usercode, will be permitted to execute restricted commands.

See also:

[RESTRICT command](#)

FOR TAPELIB

The TAPELIBUPDATER process requires a usercode assigned as the TAPE LIBRARIAN. This is DEFINEd using this option of the USE command. It must be preceded by a FOR <PU>/<password> clause although the TAPELIB USE <usercode> does NOT have to be privileged and normally isn't. (See the FOR modifier for details.)

```
FOR  MYPU/PW USE USER META FOR TAPELIB
```

USE NOPASSWORDS

NOPASSWORDS can only be set by a FOR usercode that has both PU and SECADMIN privilege.

A modifier of '+' or '-' must be specified with the command:

```
FOR META/PW USE NOPASSWORDS +
```

If this NOPASSWORDS has been set, additional text will appear in the USE response:

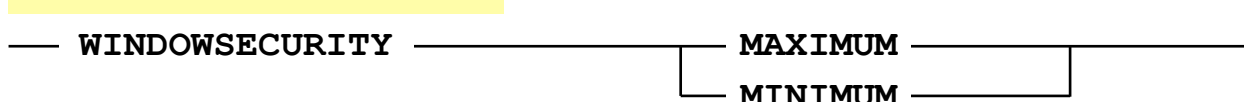
```
PASSWORD VALIDATION: NOT ACTIVE
```

This use option is used in conjunction with the Userdata attribute NODEFAULTUSE. It controls whether a password is required in a 'FOR <user>' prefix.

[See Password handling with NODEFAULTUSE](#)

USE WINDOWSECURITY

<WINDOWSECURITY spec>



WINDOWSECURITY can currently have one of two values – MINIMUM or MAXIMUM (the default). On MAXIMUM, the station must be logged on under a PU or SYSTEMUSER usercode.

```
USE WINDOWSECURITY MAXIMUM
```

If the variable is set to MINIMUM, Supervisor will not perform any checks on the usercode and log-ons to Supervisor windows will be controlled entirely by the standard COMS security mechanisms. Changing the level of WINDOWSECURITY will not affect current window sessions but will be used to validate any new log-on attempts.

```
TT USE
TT USE QUEUE 5
TT FOR PRIV/PASS USE USER OPSUP
TT USE JOB *UTIL/GREEN/LOG ON SYSTEM AFTER TL
```

The diagram shows a horizontal line representing a window. On the left, the word "WINDOW" is written. To its right, a vertical line descends from the horizontal line, followed by a question mark "?". Further to the right, another vertical line descends from the horizontal line, followed by a horizontal line segment. Below this segment, the text "<station name>" is written. A final vertical line descends from the end of the horizontal line segment.

performs a 'WINDOW- <station name>' for all active stations.

WS _____

Page 129

about the particular codefile in use. Upon receipt of a WS command, SUPERVISOR displays a page of information describing the version of the SUPERVISOR, the system serial number, the version of ATtributes in use, and compile-time options.

NOTE that the format of dates shown in the WS output conforms to the current setting of the SUPERVISOR USDATES option.

Example

```
WS
----- METALOGIC SUPERVISOR Mix 4499 -----
Release      METALOGIC 2996
SUPERVISOR Version 57.570.16 Compiled at 09:32:58 on 15/10/14
Attributes Version 57.570.04 Modified at 19:10:04 on 11/06/14
OPAL         Version 57.570.02 Compiled at 09:28:59 on 15/10/14
MAGUS        Version 57.570.08
MCPs from Version 50.0 to 57.999 are compatible with this codefile

---- LICENSE DETAILS ----
Installed modules : Supervisor      Expiry date : 26/11/2014
                  TRIM             Expiry date : 26/11/2014
                  Odt Control      Expiry date : 26/11/2014

---- RUN-TIME DETAILS ----
Maximum WHEN slots : 60
Task PRIORITY      : 50
Log session        : 4511
Compile-time option DEBUG is SET
MCP commands which match Supervisor commands must be prefixed by ODT
```

SUPERVISOR will now issue a waiting entry after initialisation, or the load of a new day's Schedule at midnight, if the Metalogic licence keys for any of its modules have expired or are due to expire within the next 30 days. Keys for Supervisor, Trim and Advanced Scheduling are all checked.

The following waiting entry shows that the Advanced Scheduling module will expire in 6 days:

```
---Job---Task-Pri---Elapsed----- 1 WAITING ENTRY -----
*60353/60362 95 :04 (SUPERVISOR) (SUPERVISOR)SUPERVISOR/LICENCEKEY/WARNING
SUPERVISOR:ACCEPT:Metalogic Keys:ADVSCHED:6 days left.
```

Note that even if a module appears as EXPIRED, it will continue to operate normally until the MAGUS library is restarted.

System-related commands

This section details those SUPERVISOR commands which provide information about the system on which the software is running, as well as commands which provide a more efficient way of carrying out standard operating tasks. Many of these commands are enhanced versions of similar system commands. Note that the AF symbol used in the table refers to the AFTER command, discussed in detail in [Scheduling](#).

Commands dealt with in this chapter are:

Command	Synonym	With AF ?
DBC	–	Yes
DELINK	-	Yes
FP	–	No
HO	–	No
JAMPACK	JAM	Yes
LU	–	No
PB	–	Yes
PDT	–	No
RESTRICT	-	No
??RUN	–	Yes
SHOW	SH	No
SYSDIR	-	Yes
SYSLOG	-	Yes
TL	–	Yes
VL	–	Yes

DBControl (DBC) Command

DBC	HELP	
	EXCLUDE	
	INTERVAL	<integer>
	QUIT	
	INFO	
	RESTART	<mix number>
		ALL
	STATUS	<mix number>
	STOP	<mix number>

The DBC command passes the <text> directly to METALOGIC/ DBCONTROL, if it is active. Metalogic's DBCONTROL program is used to control the memory allocation of DMSII databases and is discussed in more detail in the **Metalogic DBCONTROL Reference** manual.

Note that if DBCONTROL is active on the system, it must only be called *METALOGIC/DBCONTROL or an error message will be given. If the **TT DBC** command is given from an ODT, SUPERVISOR will automatically discard it allowing DBCONTROL itself to trap the command. Note that the **<mix number>** modifier must refer to an active DMSII database under the control of DBCONTROL.

```
TT DBC
----- METALOGIC/DBCONTROL INFORMATION -----
METALOGIC DBCONTROL LIBRARY VERSION 52.520.3 (Mixno:57006)
MAGUS VERSION 53.530.05
MONITOR INTERVAL is 120 seconds
USE 'TT DBC HELP' FOR LIST OF AVAILABLE COMMANDS
ACTIVE DATABASES CURRENTLY BEING MONITORED:

57012: (TAPELIB)METATAPELIB4
1783: (META)MAGUSGEN
```

DELINK Command

— DELINK —	
— COMS —	
— DSILIB —	
— FLEXLIB —	
— HWERRORSUPPORT —	
— JOBFORMATTER —	
— MAILLIB —	
— METANAPLIB —	
— OPALTAPELIB —	
— OPALUSERLIB —	
— PRINTSUPPORT —	
— SURELIB —	

During normal operations, SUPERVISOR will link to various system libraries, not all of them Metalogic software. For example, the MCPSUPPORT library JOBFORMATTER is used to decode the LOGTEXT attribute in the OPAL LOG context, PRINTSUPPORT is invoked whenever a PRINTS OPAL program is executed; COMSSUPPORT is used by GRINDER to support the COMS function. HWERRORSUPPORT will now delink Supervisor from the SLed HWERRORSUPPORT library.

The Metalogic libraries are FLEXLIB (access to OPAL PD attributes), OPALTAPELIB (supports TAPEDBC and TAPELABEL OPAL contexts), and OPALUSERLIB (supports the OPAL USERFN function)

The **DELINK** command enables the termination of a system library, to which SUPERVISOR is linked, without the need to quit SUPERVISOR itself. Note that for security reasons, the MAGUS, and METASECURITYLIB libraries may not be de-linked.

Also, many of these libraries do not necessarily link to the main SUPERVISOR stack

but tasks such as GRINDER, MAILHANDLER etc. If a **DELINK** occurs, it is likely that Supervisor or one of its tasks will re-link to the library in the future.

This command can be used to install new versions of Metalogic or system software, performing a SL of the library function and requesting Supervisor to terminate the existing linkage to the old copy of the library codefile.

If the de-linking is successful, SUPERVISOR will respond:

SUPERVISOR HAS DELINKED FROM THIS LIBRARY

If the library is not currently linked:

SUPERVISOR IS NOT CURRENTLY LINKED TO THIS LIBRARY

When de-linking the SURELIB library, SUPERVISOR will attempt to call an entrypoint called TERMINATESURE. SUPERVISOR expects this library to have the following simple declaration:

```
PROCEDURE TERMINATESURE;  
  LIBRARY SURELIB;
```

This library entrypoint requests the immediate termination of the SURE support library on the SURE repository system

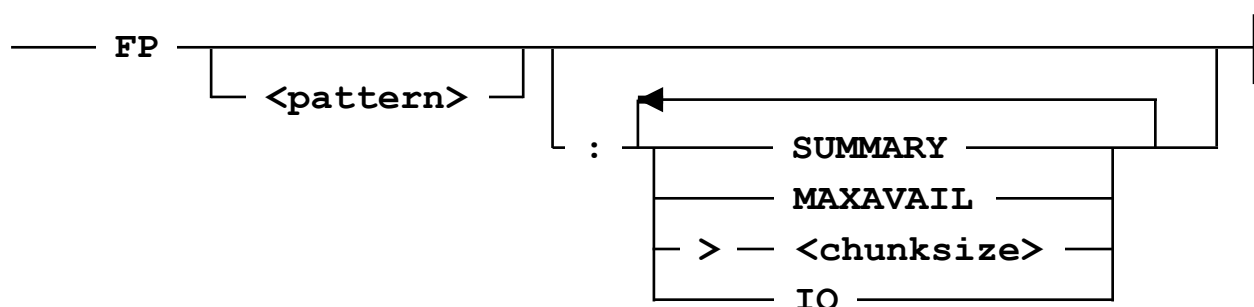
If no parameter is passed to the DELINK command the a list of currently linked libraries is returned.

```
DELINK  
  
----- Library Linkage status -----  
SUPERVISOR is currently linked to:  
  OPALTAPELIB  
  FLEXLIB  
  JOBFORMATTER  
  MAILLIB
```

Examples

```
TT DELINK COMS  
TT DELINK PRINTSUPPORT  
TT DELINK SURELIB
```

File Placement (FP) Command



The FP command interrogates the number of available segments for each disk pack

family on the system, thus giving a rapid answer to the question of which pack has the most space. The FP command performs the equivalent of the following system commands:

- PER PK to ascertain names and family indices of all disk and disk pack devices available to the MCP, and the number of files they have open.
- For each unit found above, perform a DU ON <name> (<family index>).
- For each multiple drive family found, perform a DU ON <name>.
- By default, the report lists all disks, with a summary for each family, and includes the numbers of sectors in available areas more than 504 sectors.

In the previous display, a "+" or "-" character may follow the segments figure; this indicates that the available disk space is increasing (+) or decreasing (-).

FP

Unit	Familyname	#x	DiskSize	TotAvail	%	>504	%	Open	Err
507	CDIMAGE	#1	34952554	25225330	72	25225330	72	0	0
501	DEV	#1	17476277	15102742	86	15099772	86	29	0
506	DEVAUDIT	#1	5915	1383	23	1383	23	0	0
504	DEVDLPACK	#1	5915	1887	32	1887	32	0	0
500	DISK	#1	17476277	6119387	35	6006824	34	238	0
508	DLPACK	#1	5915	1887	32	1887	32	0	0
503	DRPACK	#1	5915	1887	32	1887	32	0	0
505	LINC17	#1	5915	207	3	0	0	0	0
509	TEST5	#1	5915	1887	32	1887	32	0	0
510	TEST5	#2	5915	5887	100	5887	100	0	0
511	TEST5	#3	5915	5887	100	5887	100	0	0
	TEST5		17745	13661+	77	5887	33	0	0
** TOTALS **			69952428	46468371	66	25225330	36	267	0

FP command modifiers

<pattern>

The <pattern> modifier filters the report to those disk families whose name matches <pattern>. For example:

FP =K=

Unit	Familyname	#x	DiskSize	TotAvail	%	>504	%	Open	Err
504	DEVDLPACK	#1	5915	1887	32	1887	32	0	0
500	DISK	#1	17476277	6119387	35	6006824	34	237	0
508	DLPACK	#1	5915	1887	32	1887	32	0	0
503	DRPACK	#1	5915	1887	32	1887	32	0	0
** TOTALS **			17494022	6125048	35	6006824	34	237	0

SUMMARY

The **SUMMARY** modifier shows a similar output but lists a summary of each family only.

FP :SUMMARY									
Unit	Familyname	#x	DiskSize	TotAvail	%	>504	%	Open	Err
507	CDIMAGE	#1	34952554	25225330	72	25225330	72	0	0
501	DEV	#1	17476277	15102742	86	15099772	86	29	0
506	DEVAUDIT	#1	5915	1383	23	1383	23	0	0
504	DEVDLPACK	#1	5915	1887	32	1887	32	0	0
500	DISK	#1	17476277	6119387	35	6006824	34	237	0
508	DLPACK	#1	5915	1887	32	1887	32	0	0
503	DRPACK	#1	5915	1887	32	1887	32	0	0
505	LINC17	#1	5915	207	3	0	0	0	0
	TEST5		17745	13661	77	5887	33	0	0
** TOTALS **			69952428	46468371	66	25225330	36	266	0

<chunksize>

The "<chunksize>" modifier produces a report based on a disk area size limit other than 504 sectors.

FP =K=: > 2048									
Unit	Familyname	#x	DiskSize	TotAvail	%	>2048	%	Open	Err
504	DEVDLPACK	#1	5915	1887	32	0	0	0	0
500	DISK	#1	17476277	6119387	35	5946925	34	237	0
508	DLPACK	#1	5915	1887	32	0	0	0	0
503	DRPACK	#1	5915	1887	32	0	0	0	0
** TOTALS **			17494022	6125048	35	5946925	34	237	0

MAXAVAIL

The **MAXAVAIL** modifier reports the largest available contiguous area for each family.

FP :MAXAVAIL									
Unit	Familyname	#x	DiskSize	TotAvail	%	MaxAvail	%	Open	Err
507	CDIMAGE	#1	34952554	25225330	72	25225330	72	0	0
501	DEV	#1	17476277	15102742	86	5933736	34	29	0
506	DEVAUDIT	#1	5915	1383	23	1383	23	0	0
504	DEVDLPACK	#1	5915	1887	32	1887	32	0	0
500	DISK	#1	17476277	6119387	35	1814953	10	237	0
508	DLPACK	#1	5915	1887	32	1887	32	0	0
503	DRPACK	#1	5915	1887	32	1887	32	0	0
505	LINC17	#1	5915	207	3	207	3	0	0
509	TEST5	#1	5915	1887	32	1887	32	0	0
510	TEST5	#2	5915	5887	100	5887	100	0	0
511	TEST5	#3	5915	5887	100	5887	100	0	0
	TEST5		17745	13661	77	13661	77	0	0
** TOTALS **			69952428	46468371	66	32994931	47	266	0

IO

The **IO** modifier provides additional information about the number of I/Os for each unit and family generally since the last halt-load. The individual figures for each unit are also "reset" when a unit is un-reserved (UR-).

FP : IO

Unit	Familyname	#x	DiskSize	TotAvail	%	Reads	Writes	Mbyte	Er
507	CDIMAGE	#1	34952554	25225330	72	621	2114	547	
501	DEV	#1	17476277	15102742	86	423021	340061	15501	
506	DEVAUDIT	#1	5915	1383	23	26	1	1	
504	DEVDLPACK	#1	5915	1887	32	26	1	1	
500	DISK	#1	17476277	6119387	35	681777	1081412	7379	
508	DLPACK	#1	5915	1887	32	26	1	1	
503	DRPACK	#1	5915	1887	32	27	1	1	
505	LINC17	#1	5915	207	3	39	2	1	
509	TEST5	#1	5915	1887	32	41	4	2	
510	TEST5	#2	5915	5887	100	5	0	0	
511	TEST5	#3	5915	5887	100	5	0	0	
	TEST5		17745	13661	77	51	4	2	
** TOTALS **			69952428	46468371	66	1105614	1423597	23434	

FP may not be used with an AFTER command.

FDU information is not available for a Family when DDU INFORMATION UNAVAILABLE will be reported. This is typically caused by duplicate families on the system.

Headform Open (HIO) Command

- HO - <familyname>

- <familyindex>

<pattern>

<Filename>

The HIO (Headform Open) command reports on disk file headers, which are open on a family, even if the headers are for temporary or job files. Using the optional # <family index> syntax narrows the response to just those files which have active rows on that family index. These are the only files, which are counted in the OPENCOUNT for a non bootstrap. HIO may not be used in an AFTER command.

Each line of the response is preceded by "T" for temporary, " " for permanent, or "J" for job, followed by the open count of the file and its current size in sectors. SYSTEMDIRECTORY will always appear unless a familyindex call is used and no active directory is on the pack. Note that the SYSTEMDIRECTORY does not count in the OPENCOUNT of the PER PK system command response. Normally the OPENCOUNT on the family will be one less than the number of files in the response unless the family is the HALT LOAD unit or a DL OLAY family. In both these cases, the OPENCOUNT is increased by MCP disk structures with no disk headers.

The current file size of this file is useful for determining if a SECTORS REQUIRED condition is caused by a program looping on writes to a file, which could be temporary.

If a <pattern> is used, only the matching files will be displayed.

If the full <filename> is used, the response is a simple report detailing the tasks currently using that file.

Experienced ODT operators may note the similarities between HO and the standard ODT command SHOWOPEN. Apart from a certain improved flexibility, one subtle difference between HO and SHOWOPEN is that relative to all MCPs up to the time of writing (53.1), HO is an order of magnitude faster in execution.

Normal family interrogation (equivalent to SHOWOPEN):

```
HO DISK
```

-Tmp-	Users	Size	Name
T	1	540	*BUTTERFLY
	1	900	*COMS/CFILE
	1	27648	*COMS/INPQ
	1	1512	*COMS/TTRAIL/TPLIBRARY/0001
	1	504	*CONFIGURATION/NETWORKSUPPORT/DSS/ATTRIBUTES
	1	504	*CONFIGURATION/NETWORKSUPPORT/DSS/ENTITIES
	1	144	*CONFIGURATION/NETWORKSUPPORT/MINIMALNAME
	1	30	*CONFIGURATION/TELNETSUPPORT/VALUES
	2	1512	*DMSUPPORT/METATAPELIB4
	1	2000	*HLCN/TERMINIT/INFO
	2	1350	*JOBDESC
	1	300	*KEYEDIOII/CONTROL
	1	1000	*MDPF/ASSISTANT/STATE
	1	20916	*METALOGIC/COPYWRITE
	4	3294	*METALOGIC/COPYWRITE/BRIDGE/TCP/IP
	3	3852	*METALOGIC/COPYWRITE/HOTLINEAGENT
	4	1062	*METALOGIC/DBCONTROL
	2	7920	*METALOGIC/FLEX/LIBRARY
	2	5130	*METALOGIC/HTTP
	2	1674	*METALOGIC/MAGUS
	2	2484	*METALOGIC/MAILLIB

Window S/1 at COURSE2MCP

Filter family search to all files with the pattern "OBJECT" in the file title:

```
HO DEV =OBJECT=
```

-Tmp-	Users	Size	Name
	1	6354	*OBJECT/ED
	1	4950	*OBJECT/FLEX
	1	810	*OBJECT/MAGUS/GEN/SCREENLIB/ALL
	1	2988	*OBJECT/MAGUS/GEN

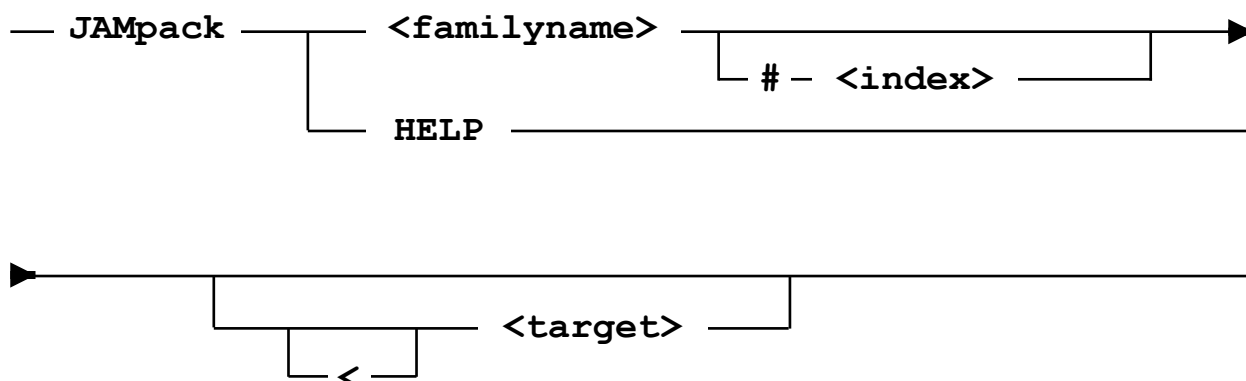
Show users of a specific file:

```
HO DISK METALOGIC/SUPERVISOR
```

----- File is in use by 6 Users -----

01007\01007	*METALOGIC/SUPERVISOR
01007\01009	(SUPERVISOR)SUPERVISOR/TAPELIBUPDATER
01007\01010	(SUPERVISOR)SUPERVISOR/SILOHANDLER
01007\01016	(SUPERVISOR)SUPERVISOR/GRINDER
01007\01017	(SUPERVISOR)SUPERVISOR/WINDOWS
01007\01054	(SUPERVISOR)SUPERVISOR/MAILHANDLER

JAMPACK Command



```

— SQUASH — <familyname> —————
                        |
                        | ( — <familyindex> — ) —

```

The JAMPACK command runs the program METALOGIC/JAMPACK (whose function is to clear contiguous areas on a pack) passing a string parameter. This can be used to handle "SECTORS REQUIRED" hangs, or to consolidate all the available sectors on a pack.

<familyname> is the family to work on.

<index> confines JAMPACK to a single family index.

<target> is the number of segments to get as a contiguous amount. If <target> is omitted, a full SQUASH is attempted.

If <target> is preceded by "<" (less than), this option says that the reserve should be first be checked by simulation. This can be expensive but gives a surer result. If the simulation fails, JAMPACK will search for the closest amount, which can be found.

HELP causes JAMPACK to print HELP information to the file LINE.

JAMPACK is a separate METALOGIC product that manages space on disk pack families. It may not be available on your system. If the utility is not resident on the system files family, SUPERVISOR will respond:

NOT DONE: REQUIRED FILE NOT RESIDENT

The SQUASH command syntax is a duplicate of the SQUASH system command and is provided for compatibility. It initiates JAMPACK in the "consolidate all sectors" mode. The pack should normally not be in use during a SQUASH run.

JAMPACK will start a RESERVE to do the data movement, and will wait for an AX. If the RESERVE completes without error, it will re-start on its own. The only valid response is "<mixno> AX QUIT". In a full squash run, JAMPACK will wait until the RESERVE completes before terminating. If a second QUIT is entered it will DS the RESERVE. Sectors required and simulations will DS the RESERVE after the first "AX QUIT" is entered.

Only one JAMPACK can be active at any one time, but a separate task is provided so there can be no contention with other utilities.

Examples

To solve a "1276 SECTORS REQUIRED ON PACK" RSVP

JAM PACK 1276

If you wished to schedule a daily run of JAMPACK to make a large area on one family member:

AF 0300 DAILY JAM CANDEPACK #1 < 100000

To schedule a full weekly SQUASH:

AF 0300 ON SUNDAY JAM SYSTEMPACK

Linked Users (LU) Command

— LU ————— <mix number list> —————

The LU command shows the tasks that are linked to a LIBRARY or DBS. The <mix number list> must contain the mix numbers of one or more library or database tasks, as seen by the LIBS or DBS system command. The response from SUPERVISOR displays the mix numbers and names for all the tasks directly using the specified mix numbers.

LU may not be used in an AFTER command.

Examples

Libraries

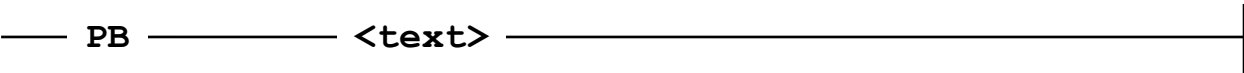
```
LIBS NAME =META=
---Mix---Frz---Shr---Usr----- 16 FROZEN LIBRARIES (ALL) =META=-----
1785 Temp All 2 Job (BOB) (META)DMSUPPORT/MAGUSGEN ON DEV
1786 Ctrl All 1 (BOB) (META)DMSUPPORT/MAGUSGEN ON DEV
1011 Temp All 2 SL Job *METALOGIC/OPALTAPELIB4
1007 Ctrl All 0 Job (SUPERVISOR) *METALOGIC/SUPERVISOR
61843 Ctrl All 1 Job *METALOGIC/SUPERVISOR/TTINTERFACE
57104 Perm All 1 SL Job *METALOGIC/FLEX/LIBRARY
50367 Ctrl All 0 SL Job *METALOGIC/HTTP
1828 Ctrl All 0 SL Job *METALOGIC/SUPERVISOR/RECORDER
1824 Conn Priv Job (IPP) *METALOGIC/COPYWRITE
57068 Ctrl All 1 SL Job *METALOGIC/MAILLIB
1821 Conn Priv Job (BOB) *METALOGIC/COPYWRITE/HOTLINEAGENT
57037 Temp All 1 SL Job *METALOGIC/TAPEMANAGER
57013 Temp All 2 Job (TAPELIB) (TAPELIB)DMSUPPORT/METATAPELIB4
57015 Ctrl All 1 (TAPELIB) (TAPELIB)DMSUPPORT/METATAPELIB4
57006 Ctrl All 0 Job *METALOGIC/DBCONTROL
57004 Perm All 9 SL Job *METALOGIC/MAGUS
```

```
LU 57004
----- LINKED USERS -----
----- MIX NUMBER 57004 IS LINKED TO 9 STACKS -----
57006/01784 *METALOGIC/DBCONTROL/META/MAGUSGEN
01011 JOB *METALOGIC/OPALTAPELIB4
01007 JOB *METALOGIC/SUPERVISOR
57104 JOB *METALOGIC/FLEX/LIBRARY
01824 JOB *METALOGIC/COPYWRITE
57068 JOB *METALOGIC/MAILLIB
57006/57014 *METALOGIC/DBCONTROL/TAPELIB/METATAPELIB4
57008 JOB *METALOGIC/SUPERVISOR/WAITWATCHER
57006 JOB *METALOGIC/DBCONTROL
```

```
ODT DBS
-----Mix-Pri--Usr----- 2 ACTIVE DATABASES -----
* 1783  50      2 Job (META) (META)MAGUSGEN
 57012  50      2 Job (TAPELIB) (TAPELIB)METATAPELIB4

LU 57012,1783
----- LINKED USERS -----
----- MIX NUMBER 57012 IS LINKED TO 2 STACKS -----
01011 JOB  *METALOGIC/OPALTAPELIB4
57006/57014 *METALOGIC/DBCONTROL/TAPELIB/METATAPELIB4
----- MIX NUMBER 01783 IS LINKED TO 2 STACKS -----
57006/01784 *METALOGIC/DBCONTROL/META/MAGUSGEN
01781/01782 *OBJECT/MAGUS/GEN ON DEV
```

Print Backup (PB) Command

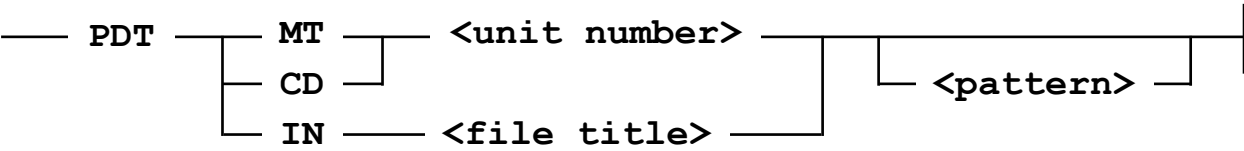


The PB command runs SYSTEM/BACKUP with the same parameter that an equivalent WFL PB statement would use. The standard ODT/WFL system command (?PB) builds a JOB, runs the WFL compiler to make a job code file, and puts it into a QUEUE. If the system or queue MIXLIMITs are not exceeded, the job is started. The job hangs if BACKUP is not on DISK (unless an appropriate UQ is set up – but a UQ has problems with starting other jobs). In short, the standard BACKUP is inefficient and has practical problems.

The SUPERVISOR PB command uses a PROCESS to go into immediate execution and avoid a JOB's memory overhead. BACKUP is run from the family specified in the USE FAMILY FOR SYSTEM and has the privileges of the USE USER usercode. The <text> parameter is passed to SYSTEM/BACKUP without analysis. For the syntax of <text>, refer to the **Unisys System Software Utilities Reference Manual**.

SUPERVISOR can only run one SYSDIR, SYSLOG, or PB command at a time. If another of these commands is active, SUPERVISOR will give an error message. If the PB comes from a scheduled activity, it is not rejected, but delayed with a postscript of TASK QUEUED.

Tape Directory (PDT) Command



The PDT command invokes SUPERVISOR's fast tape directory routine, returning a list of the files found on a LIBRARY/MAINTENANCE tape, or the header of a MEMORY/DUMP tape. The PDT command is also capable of providing directories of CD ROMs that are in Library Maintenance format.

A single unit number of a magnetic tape drive or CD ROM drive may be specified in each PDT command. SUPERVISOR will return a list of all files found on a library maintenance tape to the requesting console, and the header of a MEMORY/DUMP tape. Only these two types of tape are allowed. Any reel of a LIBRARY/MAINTENANCE tape volume may be read independently. If a <pattern> is specified, then SUPERVISOR will return only those files on the tape whose names match the pattern provided.

```
TT PDT MT 36 *=WFL=
```

```
----- SUPERVISOR PDT FOR MT 36 METASYS432A/FILE000 [RFT2  ] OF
07/11/1996 -----
----- USING PATTERN *=WFL= -----
00010 *DATABASE/WFL/AUTOREORG [KEY=432-DMSII-DM2, VALID]
00011 *DATABASE/WFL/COMPILEACR [KEY=432-DMSII-DM2, VALID]
00012 *DATABASE/WFL/COMPILEDDB [KEY=432-DMSII-DM2, VALID]
00013 *DATABASE/WFL/COPYAUDIT [KEY=432-DMSII-DM2, VALID]
00014 *DATABASE/WFL/RECONSTRUCTFILTER [KEY=432-DMSII-DM2,
VALID]
```

The Unisys standard TAPEDIR program, SYSTEM/FILEDATA, is not very user friendly when run from an ODT. For example, it does not respect the TERM specification and it does not work from a REMOTESPO.

The SUPERVISOR PDT command is an improved TAPEDIR to the ODT, replacing the standard "TDIR SPO" command. PDT is much faster, returns the response like any other system command, does not clear the screen, keeps the tape open a minimum time, and does not have to fight its way through the system job QUEUES.

Like SYSTEM/FILEDATA, PDT can read the keys information on a Unisys keyed tape.

The PDT command also has the ability to read the label of a MEMORY/DUMP tape and report back the first page information much more cheaply than running DUMPANALYZER.

An elapsed time limit of 2 minutes is applied to the internal process initiated in response to a PDT command. This limit allows SUPERVISOR to continue in case a tape drive hangs. (In rare cases, location of the directory on a continuation reel may take more than 2 minutes and in these cases PDT may not terminate correctly.)

The internal process is called TAPEDIR and is active only as long as is necessary to read the directory file from a library tape.

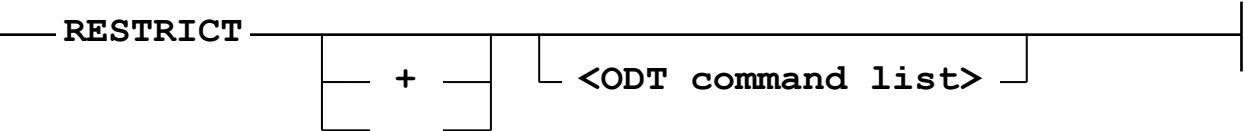
The IN form allows the contents of a CopyWrite image file or a 'LIBMAINTDIR' directory to be displayed. The ON part of the title can specify a CD or familyname. The file titles usercode and familyname, if not supplied, are modified with the USE.

EXTERNAL specifications. To use this feature for a copywrite image, the Metalogic COPYWRITE product must be previously installed.

Example

```
PDT MT 16
PDT CD 123
PDT MT 15 (META=)=OPALS=
PDT IN WRAPPED/FILE ON DEV (=)##DEV=
```

RESTRICT Command



The RESTRICT command (not to be confused with the Unisys INFOGUARD equivalent) is used to specify the ODT commands that cannot be executed without special privileges. To restrict or un-restrict ODT commands, RESTRICT can only be input from the ODT Security usercode, either with a FOR, or from a TERM USER ODT or Supervisor window. A valid ODTCONTROL license key is required from Metalogic. The ODT Security usercode is set using the Supervisor USE command and is discussed in Chapter 4.

```
TT RESTRICT

--- TT RESTRICT Command List ---

MEMORYCEILING      BNAVERSION      SEGARRAYSTART      ASD
SL                  MP
```

Unlike most other Supervisor commands, a TT prefix is mandatory when RESTRICT is used from a Supervisor window or Remotespo because of the conflict with the Unisys RESTRICT command.

When RESTRICT is used for inquiry purposes i.e. with no modifiers, the command is available to all users.

The command syntax is shown below; note that the ODT Security usercode is specified as ODTSUP, with a password of ODTSUP:

```
TT FOR ODTSUP/ODTSUP RESTRICT PG,OP,DS

--- TT RESTRICT Command List ---

MP                  SL                  ASD
SEGARRAYSTART
BNAVERSION          MEMORYCEILING      DS                  OP
PG
```

The command list can contain any operator command that is implemented via the DCALGOL SETSTATUS intrinsic, including mix and unit requests. This means that significantly more commands are available in comparison to the Infoguard RESTRICT ODT command.

As stated earlier, use of the TT RESTRICT command requires a valid ODTCONTROL license key; the validity and expiry of the key is displayed in the response to the Supervisor TT WS command.

When a command is present in the RESTRICT list, it is disallowed from anywhere other than the ODT Security usercode. The calling program, MARC user or operator at an ODT will get the error:

```
MP SYSTEM/DUMPALL +PU
COMMAND IS RESTRICTED
```

Any such restricted command is logged to the SYSTEM/SUMLOG as a failed command and will be seen in a LOG OPERATOR report. The failed command can also be detected by Supervisor, using the OPERATOR=SETSTATUS object, and will be passed to any Supervisor SITUation running with the OPERATOR context. If the OPAL is running with a FOR modifier specifying the ODT Security usercode, its ODT statements (but not KEYINs) will automatically by-pass the RESTRICT. This allows the OPAL programmer to be highly selective in controlling ODT input.

Restricted commands can be input manually from a SUPERVISOR COMS window with a TT TERM USER ODT security assigned, DCKEYIN or SETSTATUS programs running under ODTSUP or MARC systemuser sessions also active under ODTSUP.

Using RESTRICT with either '+' or '-' but **without** a command list, allows the suspension or resumption of RESTRICT enforcement without modifying the list of commands. To temporarily disable the current list of restricted ODT commands:

```
TT FOR ODTSUP/ODTSUP RESTRICT -

    --- TT RESTRICT Command List (SUSPENDED) ---

    MP                SL                ASD
SEGARRAYSTART
    BNAVERSION        MEMORYCEILING
```

The specified ODT Security usercode need not have PU or SECADMIN privileges in the USERDATAFILE, but it is strongly recommended that the usercode is restricted for use to the Security Administrator or similar.

Table of RESTRICT capable ODT commands

The following tables, sub-divided into System, Peripheral and Mix, show the MCP commands controllable by RESTRICT. Note that any interrogation variant of these commands is NOT affected.

System commands		
ACCOUNTING	AD	AI
ARCCOPY	ARCDUPLICATE	ARCREPLACE
ASD	AUTORESTORE	BNAVERSION
CF	CM	COMPILETARGET
COPYCAT	CS	DD
DF	DL	DN
DR	DRC	FILELOCKTLIMIT
HLUNIT	HN	HS
HU	ID	LC
LIBTRACE	LOGGING	MA
MAX	MB	MDT
MEMORYCEILING	MP	MU
NETEX	OP	OPLOCKTIMEOUT
PB	PP	PRIMITIVE
RA	RB	RECONFIGURE
RECOVER	RES	RP
SB	SBP	SECOPT
SEGARRAYSTART	SF	SI
SL	SQUASH	SS
SUPPRESSWARNING	SYSTEMLANGUAGE	TL
TR	XD	

Mix commands				
AX	BADFILE	DO	DS	DUMP
FA	FM	FR	FS	HI
IL	LG	LJ	LP	NOTOK
OF	OK	OU	PR	QT
RESTART	RM	SM	ST	SW
THAW	UL			

Peripheral Commands				
ACQUIRE	FORM	FREE	HOLD	IOTIMER
LB	LH	MIRROR	MODE	MOVE
PG	PGL	POWER	RC	REPLACE
RESTRICT	RW	RY	SCAN	SEND
SN	SNL	SR	SV	UR

RUN Command

—— ??RUN ————— <file title> —————|

Programs such as MCSes should be run by using a primitive RUN (??**RUN**) from a system console. SUPERVISOR allows a primitive run to be done from an AFTER command, causing an ALGOL RUN to be done on the specified code file.

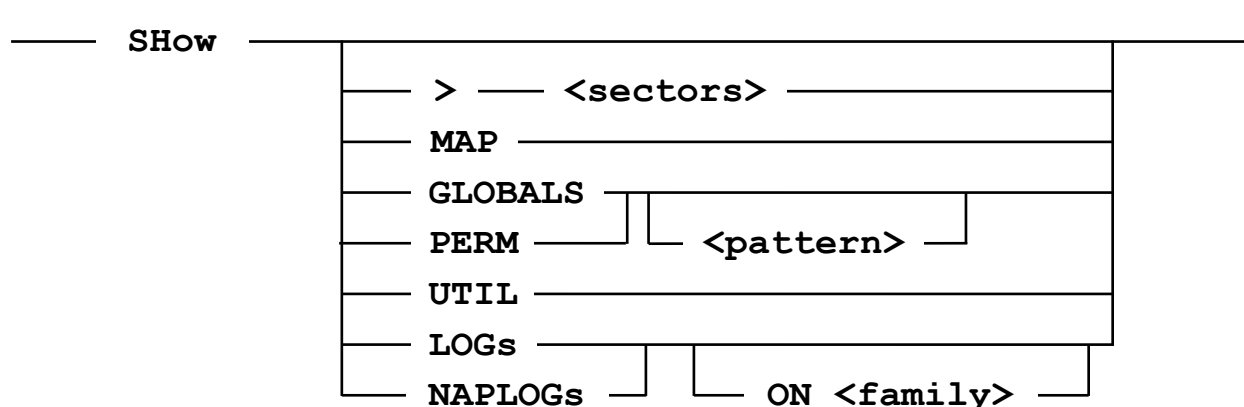
The ??**RUN** command can *only* be used as part of an AFTER command.

The same function as a ??**RUN** command can be achieved in OPAL by putting "??**RUN**" as the first five characters of the ODT input from an ODT statement.

Examples

```
TT AF + HL DAILY ??RUN SYSTEM/CANDE ON SYSTEM
TT AF 800 ON WORKDAYS ??RUN GEMCOS/MCS ON HOGPACK
```


SHOW Command



The SHOW (SH) command has a variety of modifiers, which give useful information about SUPERVISOR, SCHEDULE file, SUMLOGs or NAP logs.

When no option is present, SHOW depicts the size of the SCHEDULE file together with the space available is returned along with a summary showing the areas in use by, and numbers of SITUations, ODTSequences and MEMOs.

A default 'chunk size' of 10 sectors is used when reporting available space but this may varied by using the '>' option e.g. to report space available in chunks of 20 segments or greater:

```

SH > 20
----- SUPERVISOR SCHEDULE STATISTICS -----
SCHEDULE VERSION = 17
SCHEDULE IS 180000 SEGMENTS - 77% AVAILABLE

AVAILABLE = 140147, LARGEST AREA = 137796
1170 SEGMENTS IN 398 AREAS LESS THAN 20
138977 SEGMENTS IN 20 AREAS LARGER THAN 19

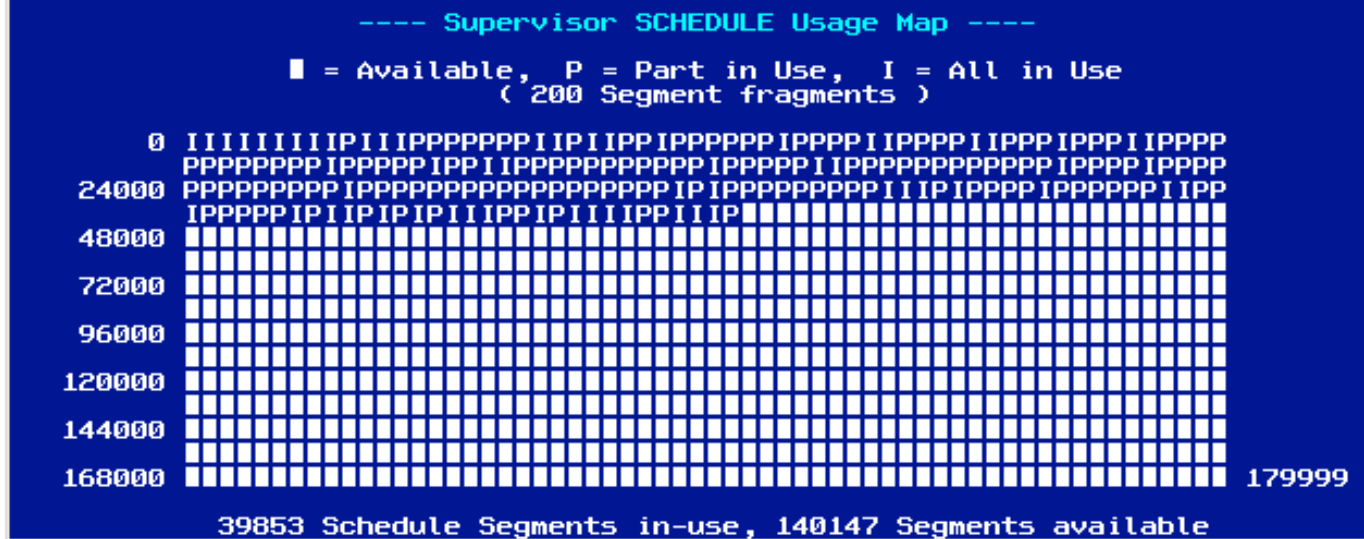
Total number of DEFINES in SCHEDULE is 3570
 2551 Segments in use by   804 SITUATION(s)
31223 Segments in use by  2317 ODTs(es)
 3228 Segments in use by   449 DISPLAY(s)
  189 Segments in use by    25 MEMO(s)
  189 Segments in use by    14 COMMAND(s)

Last METATAPELIB timestamp was 03:16:17 16/04/2008
LOG evaluations will time slice after 200 log records

```

MAP

The SHOW MAP variant gives the overall disk usage of the SCHEDULE file. This display shows the usage of the file in 200 segment fragments (the maximum size being 180000 segments) and indicates whether each fragment is completely free, partially in-use or fully utilised.



GLOBALS

This variant of the SHow command will return a list of Supervisor Global variables, with their current values. The use of <pattern> allows the request to be filtered for a specific name or wild-card pattern.

```

SH Globals
      ---- Opal Global Variables ----
      -- RECE0J (3) --
00000000:ALL
      -- RECMSG (7) --
00000000:DISPLAY
      -- RECPKSPACE (1) --
00000000:1
      -- RECSEC (3) --
00000000:YES
      -- RECSTATFAM (16) --
00000000:DEV,DISK,CDIMAGE
      -- RECSTATUS (2) --
00000000:60
      -- RECSUPURL (27) --
00000000:metallogicuk.dyndns.org:8080
      -- RECW (3) --
00000000:YES

      --- Dump of Global Real variables ---
REC_MAXHOST      000000000000A      Value = 10
RECSECS          000000000003C      Value = 60

```

With a pattern

```

SH Globals =SEC=
      ---- Opal Global Variables ----
      ---- Searching for '=SEC=' ----
      --- Dump of Global String variables ---
      -- RECSEC (3) --
00000000:YES

      --- Dump of Global Real variables ---
RECSECS          000000000003C      Value = 60

```

PERM

This variant of the SHow command will return a list of all active Magus Permanent variables with their values. The use of <pattern> allows the request to be filtered for a specific name or wild-card pattern.

Any CR (carriage-returns) embedded in a permanent string variable will be shown as the '^' character to allow easy identification.

```

TT SH PERM
      --- Dump of Permanent Real variables ---
DUMMYREAL      26C88AEFB2AB      Value = 0.56674
MCP            000000059B00C      Value = 5877772
N1             000000000237      Value = 567
T501           2093F561699A      Value = 10715147571.3
T502           2093F561699B      Value = 10715147571.4
T503           2093F561699C      Value = 10715147571.5
T504           2093F561699D      Value = 10715147571.6
U501           000000002DEF1      Value = 188145
U502           0000000794BD0      Value = 7949264
U503           000000009AD2      Value = 39634
U504           000000F0974B      Value = 15767371
X              000000000000A      Value = 10

      --- Dump of Permanent String variables ---
-- A(21) --
00000000:One=1:-Two=2:-Three=3
-- AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA(1) --
00000000:A

```

With a pattern

```
SH PERM =BOB=

      ---- Searching for '=BOB='----
      --- Dump of Permanent Real variables ---
BOBTST      000000003039      Value = 12345

      --- Dump of Permanent String variables ---
-- BOB(29) --
00000000:xy1xy2xy3xy4xy5xy6xy7xy8xy9xy
-- BOBTST(4) --
00000000:TEST
```

UTIL

The UTIL modifier will report details of Supervisor CPU utilisation, broken down into categories. It is beyond the scope of this document to describe these categories here but the values returned are useful for Metalogic personnel to investigate possible SUPERVISOR performance problems.

```
SH UTIL

      ---- SUPERVISOR PROCESSOR UTILISATION ----
SUPERVISOR was started at 1:46 PM Thursday, April 10,2008
SUPERVISOR active for      141:46:13.161
Total Processor Time      15:47.317      <1% of Active Time

  Initialisation      0.694      <1% of Total
  Scheduled Activities      6.890      <1% of Total
  Controller Queue handling      25.456      2% of Total
  Primary Queue      0.002      <1% of Total
  Exception Event      0.116      <1% of Total
  WHEN Queue      10:51.177      68% of Total
  Status Change handling      55.766      5% of Total
  Command Input processing      4.394      <1% of Total
  Supervisor Logging      0.000
  Process invocation      28.036      2% of Total
  Process Switching      2:54.765      18% of Total
  Misc Event handling      0.015      <1% of Total
  Other processes      5:40.143

      --- Accumulated WHEN CPU usage ---
Used by Active WHENs      12:23.844      78% of Total
Used by Completed WHENs      33.223      3% of Total
```

The above utilisation figures are only maintained internally if the SUPERVISOR option NOSTATS has been set. Since there is a CPU overhead associated with maintaining the above performance figures internally, especially important on Unisys metered COD systems, setting NOSTATS will disable monitoring.

```
SH UTIL

NOT AVAILABLE DUE TO SO+NOSTATS OPTION
```

LOGs

Useful information about the current SUMLOG and earlier, released logs can be extracted by using the SHOW LOGS command:

```

----- SUPERVISOR LOG SEARCH -----
Searching on DISK (D) and on DEV (B)

D *SYSTEM/SUMLOG          23:59 on 13/04/08 to Current Time/Date
D *SUMLOG/1/041308/000574 22:23 on 10/04/08 to 23:59 on 13/04/08
B *SUMLOG/1/041008/000573 22:30 on 09/04/08 to 22:23 on 10/04/08
B *SUMLOG/1/040908/000572 20:02 on 08/04/08 to 22:30 on 09/04/08
B *SUMLOG/1/040808/000571 15:59 on 08/04/08 to 20:02 on 08/04/08

```

On receipt of the request, Supervisor will attempt to locate the current SYSTEM/SUMLOG on the DL LOG Family. If successful, the sequence number of the current log file is extracted and SUPERVISOR will search for earlier SUMLOGs on the family specified by USE FAMILY... FOR LOGS.

Details of the time range for each SUMLOG found (including the current SYSTEM/SUMLOG) will be displayed: missing SUMLOGS will be skipped but their sequence numbers will be shown.

A search for previous SUMLOGs may be carried on an alternative family by specifying the family name after the optional 'ON' part. In either case Supervisor must be able to locate and open the current Log before the search can be done.

Similarly, the SHOW NAPLOGs command will provide information about resident NAPLOG files on the family specified USE FAMILY.. FOR NAPLOGS.

```
sh naplogs ON DEV
```

```

----- SUPERVISOR NAPLOG SEARCH -----
Searching on DISK (D) and on DEV (B)

D *NAPSYSTEM/NAPLOG      10:08 on 17/02/00 to Current Time/Date
B *NAPLOG/7841            02:24 on 30/06/97 to 03:30 on 30/06/97
B *NAPLOG/7840            02:16 on 30/06/97 to 02:24 on 30/06/97
B *NAPLOG/7839            01:54 on 30/06/97 to 02:16 on 30/06/97
B *NAPLOG/7838            01:39 on 30/06/97 to 01:54 on 30/06/97
B *NAPLOG/7837            23:20 on 29/06/97 to 01:39 on 30/06/97
B *NAPLOG/7836            22:49 on 29/06/97 to 23:20 on 29/06/97
B *NAPLOG/7835            16:38 on 29/06/97 to 22:49 on 29/06/97
  ** Could not locate Log Sequence Nos 7834 to 1367 **
B *NAPLOG/1366            10:37 on 22/06/02 to 20:13 on 22/06/02

```

SYSDIR Command

```

_____ SYSDIR _____ <optional text> _____

```

The SYSDIR command runs SYSTEM/FILEDATA with the same parameter that an equivalent DIR system command would use.

The standard DIR system command builds a JOB, runs the WFL compiler to make a job code file, and puts it into a QUEUE. If the system or queue MIX limits are not exceeded, the job is started. The job hangs if FILEDATA is not on DISK (unless an appropriate UQ is set up – but a UQ has problems with starting other jobs). In short, the standard DIR is inefficient and has practical problems.

The SUPERVISOR SYSDIR uses a PROCESS to go into immediate execution and avoid a JOB's memory overhead. FILEDATA is run from the family specified in the

USE FAMILY FOR SYSTEM and has the privileges of the USE USER usercode. The `<optional text>` parameter (if present) is passed without inspection to SYSTEM/FILEDATA. For the syntax of `<optional text>`, refer to the **Unisys System Software Utilities Manual**.

SUPERVISOR can only run one SYSDIR, SYSLOG or PB command at a time. If another of these commands is active, SUPERVISOR will give an error message. If the DIR comes from a scheduled activity, it is not rejected, but delayed with a postscript of TASK QUEUED.

SYSDIR may be the subject of an AFTER command.

Examples

```
TT SYSDIR PACK
```

```
TT AF 0100 ON MONDAY,FRIDAY SYSDIR USERPACK
```

SYSLOG Command

—— SYSLOG —— `<optional text>` —————|

The SYSLOG command runs SYSTEM/LOGANALYZER with the same parameter that an equivalent LOG system command would use. The standard LOG system command builds a JOB, runs the WFL compiler to make a job code file, and puts it into a QUEUE. If the system or queue mix limits are not exceeded, the job is started. The job hangs if LOGANALYZER is not on DISK (unless an appropriate UQ is set up – but a UQ has problems with starting other jobs). In short, the standard LOG is inefficient and has practical problems.

The SUPERVISOR SYSLOG command uses a PROCESS to go into immediate execution and avoid a JOB's memory overhead. LOGANALYZER is run from the family specified in the USE FAMILY FOR SYSTEM and has the privileges of the USE USER usercode. The `<optional text>` parameter (if present) will be passed to LOGANALYZER without analysis. For the syntax of `<optional text>`, refer to the **Unisys System Software Site Management Reference Manual**.

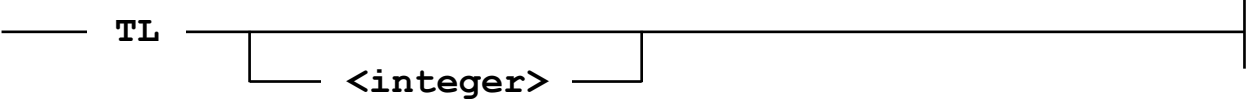
SUPERVISOR can only run one SYSDIR, SYSLOG, or PB command at a time. If another of these commands is active, SUPERVISOR will give an error message. If the LOG comes from a scheduled activity, it is not rejected, but delayed with a postscript of TASK QUEUED.

Examples

```
TT SYSLOG CON 800 MAINT MT UNSORTED FOR PRIV/PASS
```

```
TT AF 2300 DAILY SYSLOG 2000 TO 2300 SECURITY UNSORTED
```

Transfer Log (TL) Command



The TL command releases the system SUMLOG, while avoiding most of the usual operational problems. TL may be scheduled with the AFTER command. Following a TL command, SUPERVISOR:

- simulates a TL system command to release the current SUMLOG.
- copies and removes each file found in the SUMLOG directory on the family set by the DL LOG system command to the LOG BACKUP family as set by the FAMILY FOR LOGS variant of the USE command, unless they are the same. The associated *SECURITYLOG files are treated in the same way.

If the FAMILY FOR LOGS is different from the DL LOGS family and the Supervisor user code does not have the Secadmin privilege, this command will return an error.

This mechanism also applies if SO+AUTOTL is set and the SUMLOG is released manually by an ODT TL command or due to log size.

For each SUMLOG found on the DL LOG family, SUPERVISOR starts the job specified by the JOB AFTER TL option of the USE for each of the above files passing the optional <integer> as a parameter to each. The optional <integer> is provided solely for site purposes. If no value is specified, zero is used.

The job nominated as the "JOB AFTER TL" must have 2 parameters as follows.

```
BEGIN JOB ANALYSE/LOG (STRING LOGTITLE, INTEGER FUNCTION) ;
```

When SUPERVISOR starts the job, LOGTITLE will contain a file title with an "ON <family name>" part. The value of FUNCTION is the <integer> value supplied in the TL command or zero if no <integer> was specified.

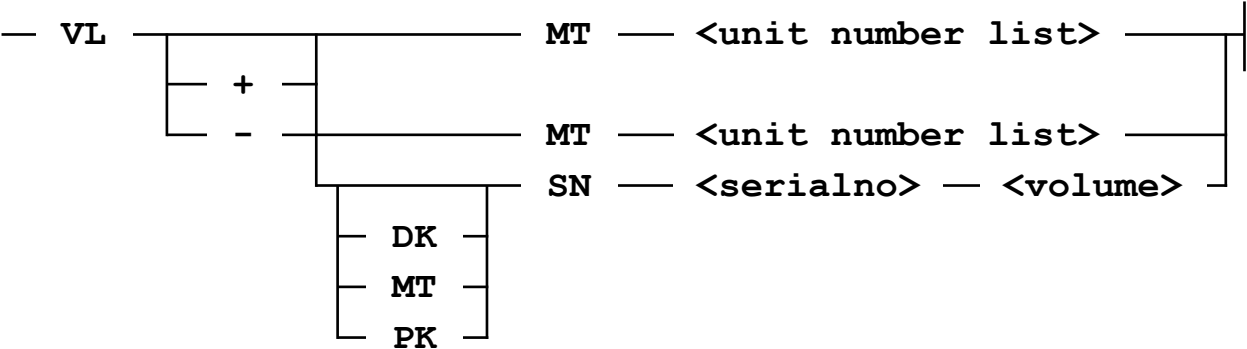
Examples

```
TT TL
TT AF 2215 ON WEEKDAYS TL 2
```

See Also:

[USE Command](#)

Volume Library (VL) Command



The VL command simplifies the process of entering tape volumes into the Volume Library. VL causes SUPERVISOR to generate and run a small job to volume add (VL+) or volume delete (VL-) on a single reel scratch or labelled magnetic tape family.

VL applies only on cataloging systems, and only to scratch tapes or labelled tapes with one reel. Checks are performed to ensure that the tape is ready, available to the group, not rewinding, not in use, labelled, and either not volumed (VL+ variant) or volumed (VL- variant).

Suitable error messages are given if the operation cannot be done. SUPERVISOR's VL command is more efficient than the corresponding WFL VOLUME statement.

A special variant of VL- allows a tape volume to be deleted if the media is not physically loaded. The VL - SN variant requires both serial number and volume name. This command may also be used for both DISK and PACK entries (e.g. FLEX CopyWrite volumes) though the modifier PK or DK must be included in the command or MT will be assumed. If the Metalogic FLEX package is installed, it is not possible to VL - a volume that FLEX knows to have backup references.

Examples:

```
TT VL + MT 15,16-19,23
TT VL - MT 15
TT VL - SN AA0001 MYVOLUME
TT VL - PK SN DF0077 ALTIMAGE_A
```

For more information on cataloging systems, please refer to the **Unisys A Series Disk Subsystem Administration and Operations Guide**.

Administration Commands

This chapter deals with SUPERVISOR's "administrative" command subset. These are the commands used to perform backups of critical SUPERVISOR files and commands relating to operations staff.

Commands	Synonym	With AFTER
ACCESS	–	No
INPUT	–	Yes
LC	-	Yes
LOG	-	No
MAIL	-	Yes
RELOAD	RL	No
REMIND	–	Only
SAVE	SV	Yes
SHIFT	–	No

The MAIL command is dealt with, in more detail, in the **Metalogic MAIL Reference manual**.

ACCESS Command

– ACCESS – <accesscode> <old password> <new password> |

The ACCESS command changes the password associated with an operator accesscode.

<accesscode> must be a valid operator accesscode previously established by the system Security Administrator and known to SUPERVISOR because of its appearance in the ACCESSCODELIST of its usercode.

<old password> must appear if a password is currently DEFINEd for <accesscode>. After use of this command, <new password> becomes the only valid password for that <accesscode>.

Example

```
TT ACCESS DUPONT/MONIQUE MICHELLE
```

See Also:

[NAMEOPERATORS Option](#)

[SHIFT Command](#)

[USE USER Command](#)

INPUT Command

—— INPUT ———— <file title> —————|

Upon receipt of an INPUT command, SUPERVISOR calls a co-routine stack called "LOAD", which opens the file specified by <file title>, begins reading records, and passes the contents to CONTROLLER where they are interpreted as system commands.

The value of <file title> must be a valid value for the File Attribute TITLE. The file can be any convenient symbolic filekind. INPUT assembles commands from columns 1-80 of files of filekind JOBSYMBOL or CDATA, and columns 1-72 of all others. All characters after a percent (%) character are treated as comment and ignored. Each input is terminated by the back slash character "\" (HEX E0), and the concatenated records are passed as one system command to the DCKEYIN intrinsic. The system command and the response are printed on a printer file.

If the file does not exist, SUPERVISOR will hang with a NO FILE or similar condition, which must be handled by the operator. This action is deliberate, as it is anticipated that INPUT will be used for queue set up, system option initialization, and similar purposes where non-performance could be disastrous. No usercode or family is assumed for the file and its title must be specified in full. The INPUT command may be the subject of an AFTER command.

SUPERVISOR commands are *not* accepted from an INPUT command file.

Examples

```
TT INPUT (OPERATIONS)QUEUE/SETUP ON SYSTEM
TT AF HL ON WEEKENDS TT INPUT *SYSTEM/OPTION/SETTINGS
```

LC Command

—— LC ———— CLOSE —————|
 | <text> |

The LC command creates a special entry in the SUPERVISOR Log file, identified by the 'Lgc' category, including the free format <Text> string in the record.

The LC (Log Comment) command recognises the CLOSE keyword which forces SUPERVISOR to close the current log and create a new file.

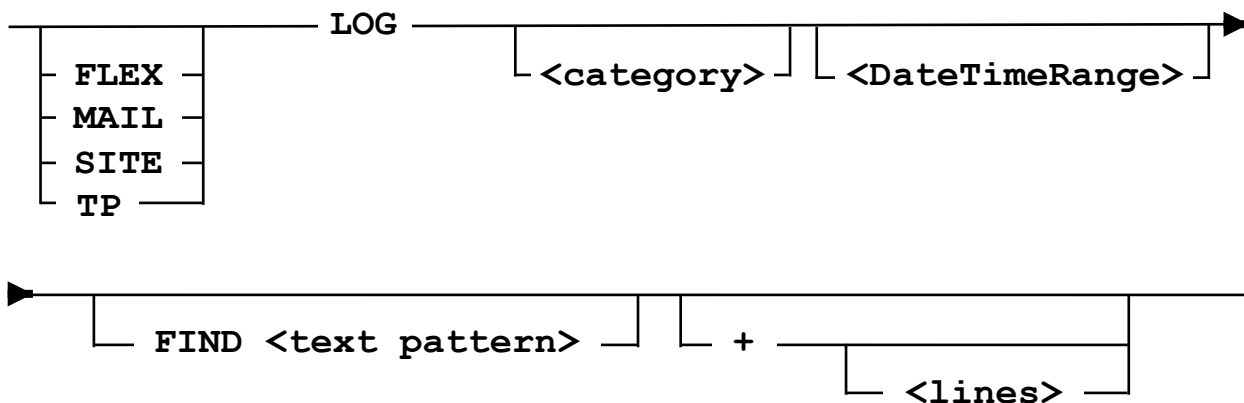
LOG Command

SUPERVISOR records all activity into a dedicated file called *METALOGIC/SUPERVISOR/LOG on a nominated family configured using the INSTALL utility. By default, this family is set to that of DL LOG.

The LOG command has been implemented to allow varied access to these log files; searching by category, wild card text patterns or time and date ranges is permitted.

MAIL LOG and TP LOG SITE LOG and FLEX LOG use the same syntax but with different categories

The LOG command syntax is described below.



<category>

All SUPERVISOR log entries are identified by a <Category> field which consists of a 3-character text string and can be used to filter any any log search by message type.

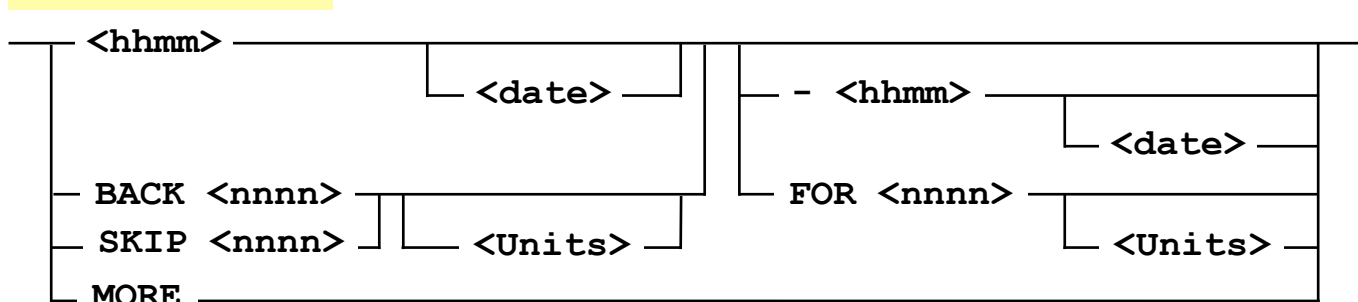
ALL	returns last 500 entries in one or more logs
ERR	returns all error messages
LGC	returns all LC (Log Comment) entries
MSG	returns all solicited and unsolicited messages
WHN	returns all scripting actions such as WHEN,Do and EVAL

By default, SUPERVISOR will only show one page of information for each of the above commands (except for ALL) but note this may not always be complete if SUPERVISOR fails to find enough entries.

A MAGUS configuration variable called SYS_LOGALL has been implemented to support a customer enhancement. This variable, supported by INSTALL's SYSTEM CONFIG menu, expects a usercode list (separated by commas) that is checked each time a LOG command is used from a logged-on remote station, typically a SUPERVISOR window.

If the logged-on usercode is in the list, SUPERVISOR appends '+' to all LOG commands (unless '+' is already present) so that each command can return more than one page of information. Any changes to the usercode list assigned to SYS_LOGALL will require a SUPERVISOR restart. This feature applies for all LOG commands for SUPERVISOR, TRIM and MAILLIB.

<DateTimeRange>



The <date> format depends on the global setting of the Metalogic configuration variable SYS_USDATES which can be set with the INSTALL utility. If SYS_USDATES is set to TRUE then <date> has the form mm/dd/yy, otherwise dd/mm/yy.

The MORE modifier set the Stop date and time to be the date and time of the last log record reported in the most recent log command.

The BACK and SKIP modifiers respectively allow specific start and end time points to be more easily assigned without having to use explicit values. BACK allows a simple log start time to be assigned whereas SKIP assigns a log End time. The FOR specifies an optional duration associated with the Start or End time. If FOR is omitted then both BACK and SKIP will assign default search range periods of 24 hours.

<Units>



By default, MINUTES will be used if no <Units> is given. A minimum of one character may be used to specify any <Units> type. BACK and FOR may be used in any combination with normal date and times.

If the '+' modifier is used, an optional count may be used to control the number of response lines returned to the caller. This modifier has a default of 500 lines and should be used with care if a filter is used with many logs present.

'+' cannot be used as part of a <Text Pattern>

To search for the literal characters # @ & = ~ they should be prefixed by the \ character. Ex. LOG FIND \= to find the literal "=" in a log entry.

If a <DateTimeRange> is specified or the '+' modifier is used in the command text, then all available log files will be searched until a maximum of 500 lines of text has been returned.

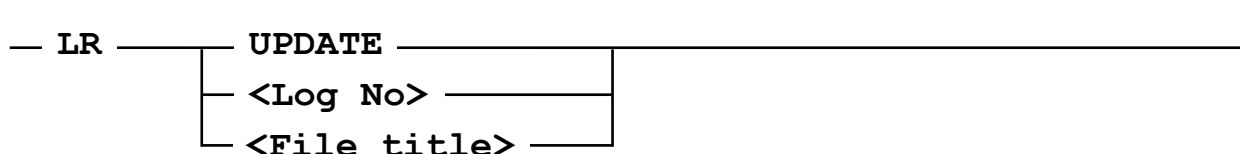
The FIND modifier permits the searching of each log entry for the specified text. The FIND automatically encloses the target with '=' at both ends of the text and is NOT case-sensitive, therefore finding both lower-case and upper-case matches.

FIND must be used after all other modifiers as spaces are permitted with in the FIND text. The log files are automatically released when the file size exceeds 15,000 records or the LC CLOSE command is used.

Examples

```
LOG BACK 2 H FOR 10 MIN
LOG BACK 20 FIND TERMINATION
LOG 0800 1/1/07 FOR 2 DAYS
```

LR Command



The LR command is the interface to the SUPERVISOR/LOGREADER program, used to process tape transactions from Sumlogs if Supervisor has been down for any length of time.

The UPDATE modifier allows SUPERVISOR to request processing of the relevant SUMLOG(s) since the last known update of the TRIM database. This is implicitly performed after every SUPERVISOR restart due to a QUIT SUPERVISOR command or H/L. The latest TRIM update timestamp can always be seen in a SHOW command response.

The LR command also allows the simple processing of any resident SUMLOG file. An individual SUMLOG can be specified by using the MCP log sequence number or its full title, including family name. If a log number is specified, LOGREADER will search for the first available SUMLOG, matching the log number with the last level of the title. Both the DL family and SUPERVISOR's USE FAMILY FOR LOGS setting are searched.

If the SO LOGGING option is set, LOGREADER will write log entries with a category of 'Lgr' into SUPERVISOR's log file. This includes any error and progress messages generated by LOGREADER.

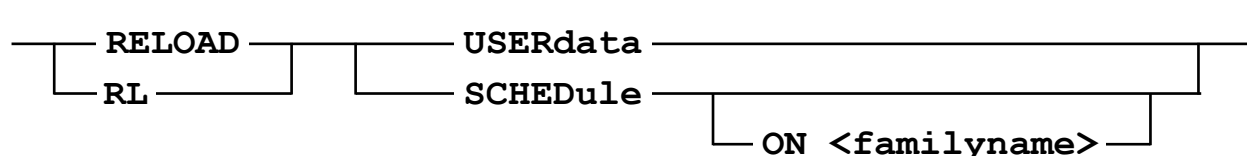
MAIL Command

MAIL	
<ABORT command>	
<ALTPortno command>	
<ALTServer command>	
<ATTACH command>	
<CODESET command>	
<CONFIG command>	
<DEBUG command>	
<DOMAIN command>	
<DUMP command>	
<HELP command>	
<IPaddress command>	
<LC command>	
<LOAD command>	
<LOG command>	
<MINLINES command>	
<MODify command>	
<NUDGE command>	
<OP command>	
<PORTNO command>	
<POSTMASTER command>	
<QUIT command>	
<REBUILD command>	
<RESUME command>	
<SERVER command>	
<SHOW command>	
<STATUS command>	
<SUSPEND command>	
<TEST command>	
<UNLOAD command>	

The MAIL command provides an operator interface from SUPERVISOR into the Metalogic MAILLIB library.

The command syntax (relative to MAILLIB version 530.01) is shown below for completeness but is subject to change from release to release. Full details of each command variant can be found in the current Metalogic documentation for the Mail system.

RELOAD Command



The RELOAD command is used to recall a SCHEDULE or USERDATA file previously backed up (manually or by a [SAVE command](#)). Upon receipt of a RELOAD command SUPERVISOR will look for, and attempt to copy a file from the 'SAVES' family (the family specified by the USE FAMILY FOR SAVES command) to the appropriate destination family. For the USERDATAFILE, this destination is the family specified in the DL system command. In the case of the SCHEDULE, it is the family specified in the TT USE FAMILY ... for Schedule command. Backup copies of the SCHEDULE may be reloaded from any family if the ON modifier is used.

RELOAD SCHEDULE looks for a backup copy of SUPERVISOR's Schedule file. (The backup can be made by a SAVE SCHEDULE command for convenience.) The title must be "*SAVED/SCHEDULE" and if no family is specified, SUPERVISOR's 'SAVES' family is assumed. RELOAD calls a co-routine named "RELOAD SCHEDULE" which fires off LIBRARY/MAINTENANCE with:

```
COPY & COMPARE SAVED/SCHEDULE AS SCHEDULE  
FROM <saves family> TO <DL JOBS FAM>
```

After a successful copy, the validity of the new file is verified in the same way as the resident SCHEDULE is checked at each SUPERVISOR initiation.

RELOAD USERDATA looks for a backup copy of the USERDATAFILE - the backup should be made by the SAVE USERDATA.

If the system SECADMIN option is set to Authorised, the usercode reported by the TT USE command as USER: must have the SECADMIN privilege set or this command will be rejected.

The file must have been saved with the name *SAVED/SYSTEM/USERDATAFILE. If the file is found, it is copied as *SYSTEM/USERDATAFILE to the DL USERDATA family. This is done by the RELOAD process, which performs the necessary freezing of the USERDATAFILE, and then invokes LIBRARY/MAINTENANCE:

```
COPY & COMPARE (<USE USER>) SAVED/SYSTEM/USERDATAFILE  
AS SYSTEM/USERDATAFILE  
FROM <backup family> TO <DL USERDATA family>
```

The necessary interaction with MCP, via the USERDATAFREEZER intrinsic, is performed by the "RELOAD USERDATA" task. For a short time while the file is being copied, the USERDATAFILE is "frozen" against changes, but interrogation is still available.

During any RELOAD process, all communication to and from SUPERVISOR is suspended.

Examples

```
TT RL USERDATA
TT RELOAD SCHEDULE
TT RL SCHED ON PACK
```

See Also:

[SAVE Command](#)

[USE FAMILY FOR SAVES Command](#)

REMIND Command

———— REMIND ————— <text> ————— |

The REMIND command is used to supply a prompt message to the operator at some time or times in the future. It is the most drastic way to try and attract the operator's attention, but can only be used together with the AFTER command. At the specified time or times, the ADM on all ODTs will be interrupted and the message together with a request to enter "TT REMINDOK" will be presented at 5-second intervals. After 5 minutes, if the operator has not responded, SUPERVISOR will assume the machine is unattended and continue; the activity will be marked with a postscript of "TIMED OUT".

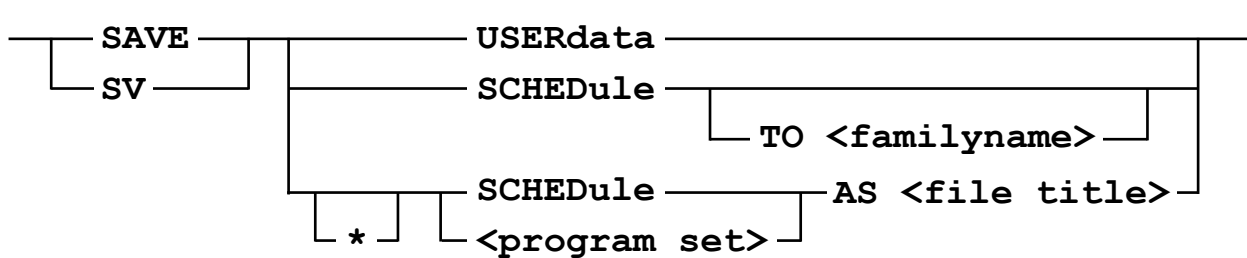
Any text prefixed by a TT which follows a valid AFTER <time specification> is not examined by SUPERVISOR when it is entered, but only when the activity is executed. In most cases, this causes no problems but if a REMIND follows the TT it will generate an invisible error at the time(s) it is scheduled.

The text of the reminder is scanned for lookup functions before it is sent. This simplifies the use of the highlighting capabilities of TD-compatible ODTs. The reminder text can be up to 19 lines of 80 characters, a total of 1520 characters. This is one full TD-screen less the 5 lines that are needed by the REMIND header, the standard messages and the default two first lines as set by the TERM FIRSTLINE system command.

Examples

```
AF SHIFT BEGIN TOMORROW REMIND PM AT 1500, SAVES AT 1400
AF 0600 ON MONDAY REMIND ADJUST AIR CONDITIONING SENSOR
```

SAVE Command



The SAVE command is used to implement a multi-level approach to error recovery. There are two forms of the SAVE command: the first form discussed here is for backing up the SCHEDULE file and the USERDATA file in their binary formats; the second form (SAVE <program set> discussed in [OPAL-related Commands](#)) is used to selectively backup OPAL programs in symbolic format.

The default location for saves is the Family defined by the TT USE FAMILY command. The TT USE command reports this family as 'Family for Saves'

Save Userdata

The SAVE USERDATA command creates a backup copy of the SYSTEM/USERDATAFILE, copying from the DL USERDATA family (refer to ODT DL command) to the 'Saves' family. Access to the Userdata file is suspended during the copy to ensure that a partially changed file is not copied.

If the system SECADMIN option is set to Authorised, the usercode reported by the TT USE command as USER: must have the SECADMIN privilege set or this command will be rejected.

Files are copied into the directory *SAVED.

The [Reload command](#) can be used to restore the Userdata file from this backup.

Save Schedule

SUPERVISOR is a continuously running program which must hold information, possibly for years in advance, which is kept in the file *SCHEDULE. The SAVE SCHEDULE command is one way of preserving the contents of the SCHEDULE file.

The SAVE SCHEDULE command makes a literal bit-by-bit copy of the SCHEDULE by doing a LIBRARY/MAINTENANCE equivalent to:

```
COPY&COMPARE SCHEDULE AS SAVED/SCHEDULE FROM <DL JOBS FAM>
TO <backup family>
```

where <DL_JOBS family> is the family specified for JOBS in the DL system command and <backup family> is the family SUPERVISOR is using for SAVES (see the USE command).

To COPY to an alternative family, the syntax SAVE SCHEDULE TO <family> can be used. The saved SCHEDULE files can be re-loaded using the RELOAD command

If SUPERVISOR's BACKUP family (i.e. the pack to which the SCHEDULE is saved)

is off line at the time of a SAVE SCHEDULE then SUPERVISOR will retry until the pack is available. A warning is issued that the BACKUP family is off line and a delay is taken between retries.

SAVE SCHEDULE AS

The SAVE AS command creates a symbolic file backup of the schedule. The created file contains all of the commands which would be entered to recreate the current schedule. The generated file is of type Jobsymbol and can therefore be edited. The [ENTER command](#) can be used to process this file.

It is also a potential emergency backup, in the case where SUPERVISOR cannot find a valid SCHEDULE on firstly the DL JOBS family, or secondly a backup copy on the 'USE FAMILY FOR BACKUP' family with the title *SAVED/SCHEDULE.

In this case SUPERVISOR will create an empty SCHEDULE and attempt an automatic rebuild before proceeding.

If the FILE FOR REBUILD has been specified in the USE command then and enter will be performed from that file. As with all ENTERs is no usercode is specified for the 'REBUILD' file then the 'USER FOR EXTERNAL' from the USE command will be used. If no family is specified then the FAMILY FOR EXTERNAL from the Use command will be used. If no FILE FOR REBUILD is specified then Supervisor will search for a file titled SAVED/SCHEDULE/DEFAULTS under the 'USE USER' usercode (normally (SUPERVISOR)) on the 'USE FAMILY FOR BACKUP' family.

For BNA environments, both the ENTER and SAVE...AS commands will allow the loading or creation of Opal files across a network by using the AT <hostname> syntax. SUPERVISOR will ensure that the remote host is visible before attempting the procedure.

Examples:

```
SV SCHED AS (SUPERVISOR)SAVED/SCHEDULE/DEFAULTS ON BKUP
SAVE SCHEDULE AS MYHOST/SYMBOL/SCHEDULE ON DEV AT YOURHOST
```

SCHEDULE conversions

In new installations of SUPERVISOR where a SCHEDULE conversion is required, the existing SCHEDULE file will now be automatically secured before the conversion procedure starts. The current SCHEDULE file will be copied to the family specified by USE FAMILY ... FOR SAVES and will be renamed as:

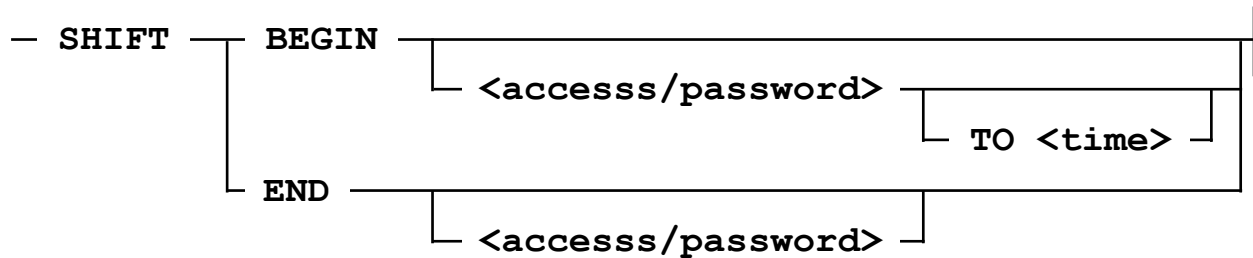
```
*SAVED/SCHEDULE/VERSIONnn_mmdyy
```

As of SUPERVISOR version 530.42, the current SCHEDULE version is 17.

The [SHOW](#) command report the current schedule version.

In the event of a SUPERVISOR release reversion after a SCHEDULE conversion, this saved SCHEDULE file can be reinstated with the old release.

SHIFT Command



The basic idea of the logging feature is that operators can inform SUPERVISOR that they are starting or ending a SHIFT. This is done with the SHIFT command to which SUPERVISOR will respond with a greeting like:

BEGINNING OF SHIFT NOTED, GOOD EVENING <accesscode>

Upon receipt of a SHIFT command, SUPERVISOR will perform any activities scheduled (with an AFTER command) for "SHIFT BEGIN" or "SHIFT END". Another SHIFT-related activity that may be scheduled is a reminder (set by the REMIND command), which displays messages on the system consoles and requires an acknowledgement. Use of these features enables one operator to leave messages for another. These activities will be immediately performed even if the SUPERVISOR Hold Schedule flag is on.

In addition, SUPERVISOR maintains a list of operators who are currently "logged on". If the LOGOPERATOR option is set, all WFL jobs will have an LJ command performed on them as they leave the system queues. The text logged contains the accesscodes of the operators who have entered a SHIFT BEGIN command. A typical message on a job summary might be:

OPERATOR ENTERED: OPERATORS AT BOJ: SMITH, JONES

If no operators have identified themselves to SUPERVISOR, the list will contain the word "NONE".

If the option NAMEOPERATORS is set, all SHIFT commands must include authorised operator identification and any invalid names will be rejected. The identification may include a password. If the NAMEOPERATORS option is reset, operator identification is optional. The password for each operator identity may be individually changed using SUPERVISOR's ACCESS command. If SUPERVISOR detects the SLED Library METASECURITYLIB, it will also perform password checking on SHIFT BEGIN commands.

If the NAMEOPERATORS option is set, SUPERVISOR will not accept most commands from the ODTs unless at least one operator has entered a SHIFT BEGIN command. Commands from SUPERVISOR windows and other sources will be accepted.

An operator is considered logged on until a "SHIFT END" command is entered with the same operator identification, or until SUPERVISOR terminates. Operator identification is not retained over a Halt Load.

The behaviour described thus far leaves two problems unresolved. The first

problem is that there is no positive reason for the operators to remember to log on. The second problem is, having logged on, there is no reinforcement to remind operators to log off.

To address the first problem, SUPERVISOR periodically stops the ADM on all system consoles if the following three conditions are met:

- SUPERVISOR's LOGOPERATOR option is set
- SUPERVISOR's NAMEOPERATOR option is set
- There are no operators logged on

The ADM is stopped (not deleted), every minute until a valid SHIFT BEGIN (together with the accesscode and its access password, if any, necessary because LOGOPERATOR Option is set) is entered. Each time the ADM is stopped, SUPERVISOR issues a message

PLEASE ENTER SHIFT BEGIN FOLLOWED BY YOUR ACCESSCODE

The effect of this action is to make it inconvenient, although not impossible, to use the system consoles until someone logs on to SUPERVISOR.

The second problem is resolved by allowing an operator to specify the length of his shift when logging on. If no duration is specified, SUPERVISOR assumes default shift duration of 8 hours. At the nominated time, SUPERVISOR simulates a SHIFT END for that operator, including any activities scheduled with an AFTER SHIFT END command.

See Also:

[ACCESS Command](#)

[AFTER Command](#)

[SO Command](#)

Defining Operator Identification

Should validation of operator identification be desired, it is only necessary to place each name in the USERDATA node ACCESSCODELIST. SYSTEM/MAKEUSER is the best way to do this. If desired, passwords may be supplied at the time the accesscodes are DEFINEd. The passwords may be changed by anyone knowing the current password of an accesscode with SUPERVISOR's ACCESS command. It is not possible to change an accesscode with a password into one without a password.

A typical use of this feature might be as follows. Suppose there are 4 operators whose names are Jones, Smith, Muller and Dupont. Someone with access to a privileged usercode must run SYSTEM/MAKEUSER to modify SUPERVISOR's usercode entry.

```
USER SUPERVISOR ACCESSCODELIST=
JONES/PW1, SMITH/PW2, MULLER/PW3, DUPONT/PW4;
```

The NAMEOPERATORS and LOGOPERATOR options could then be set:


```
TT SO + NAMEOPERATORS, LOGOPERATOR
```

From this point on, before SUPERVISOR will accept most commands, the operator must log-on, if it were Jones, with

```
TT SHIFT BEGIN JONES/PW1
```

Should Jones wish to change her password, she could enter

```
TT ACCESS JONES/PW1 MONICA
```

The list of names need not be restricted to operators, typically the system programmer of the installation might have an accesscode, the operations manager another, and so on.

Entry of SUPERVISOR's USE USER <usercode> command will cause the old usercode to be deleted from the system USERDATAFILE. All accesscodes in the ACCESSCODELIST are also lost. SYSTEM/MAKEUSER must be run to create a new list of valid operator identifications.

Examples

```
TT SHIFT BEGIN
```

```
TT SHIFT BEGIN SMITH TO 1730
```

```
TT SHIFT END JONES/PASSWORD
```


OPAL related commands

OPAL is an interpretive language customized to solving the problem of writing operational software on Unisys A Series machines. SUPERVISOR contains an OPAL compiler which is in many ways similar to other compilers on A Series systems. The compiler takes OPAL source and produces "code" for the OPAL machine – an ideal software-implemented virtual machine for OPAL's use.

A compile is done by using a form of the DEFINE command. Compiled programs are "run" using commands such as DO, EVAL, TEST, WHEN and so on. Reports can be generated using DISPlays and these can be invoked using the "/" command. Output can be re-directed using the PRINT Modifier. Sources of OPAL programs can be ENTERed from disk files and SAVED off into disk files.

For more details on the OPAL Programming Language itself, refer to the **Metalogic OPAL Reference Manual**.

Commands dealt with in this section.

Command	Synonym	With AF
DEFINE	DEF	No
DISPlay	/	Yes
DO	–	Yes
ENTER	–	Yes
EVALUATE	EV	Yes
MEMO	–	Yes
OK	–	No
ONCE	–	Yes
PRINT	PR	Yes
SAVE	SV	Yes
UO	–	Yes
WHEN	–	Yes

DEFINE Command

The DEFINE command is used to maintain the library of OPAL programs in SUPERVISOR and to compile new programs. The command has many variations and each is described in detail in the sections that follows.

The information contained in the DEFINE command is used to create the OPAL program, name it and control the execution of the compiler.

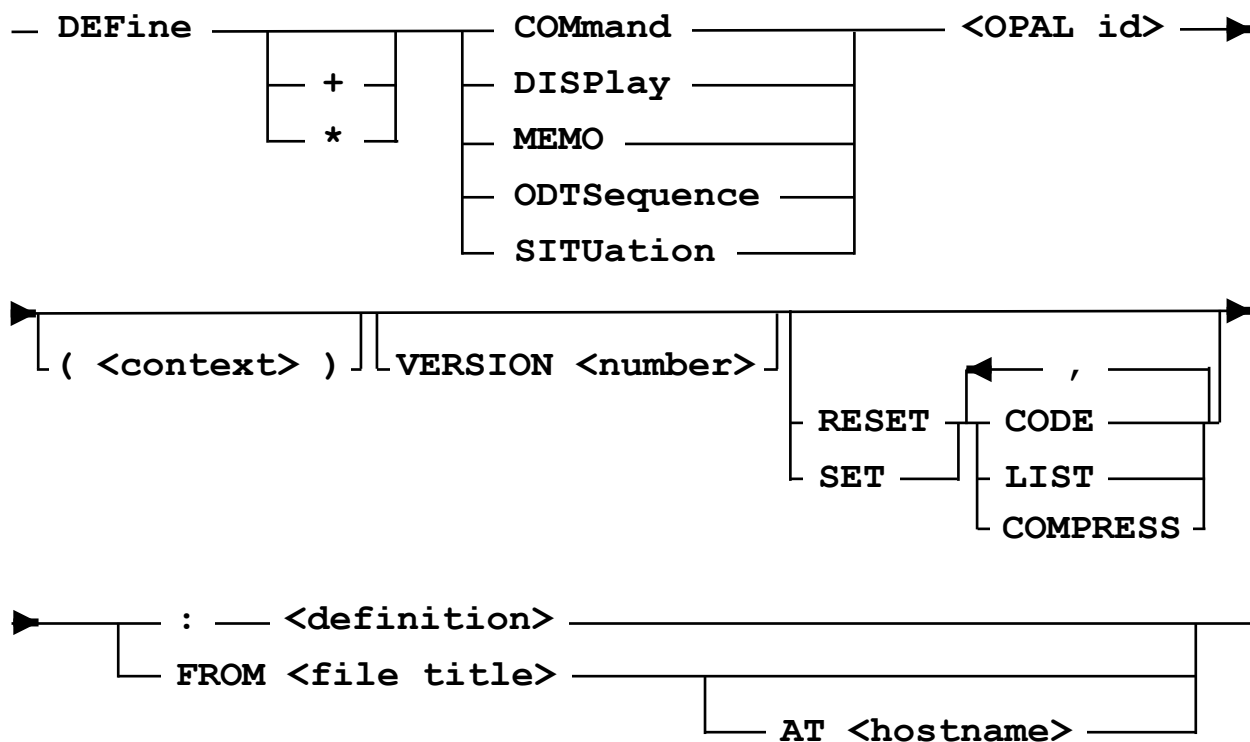
Briefly, DEFINE can:

- create OPAL Programs or MEMOs.
- interrogate the OPAL Directory held within SUPERVISOR's SCHEDULE

- list OPAL Programs or MEMOs
- modify OPAL Programs or MEMOs
- delete OPAL Programs or MEMOs

DEF + : Creating or Modifying Definitions

The general syntax for creating or modifying an OPAL program or MEMO definition is shown below:



The presence or absence of the plus character (+) immediately following the keyword **DEFINE** indicates whether this is a new program (without +) or a modification (with +). If the modification form is used, there is no check to prevent overwriting an existing program.

DEFINE + is able to define routines which are in use in an active Slot. The new code will not be used until the slot is stopped and restarted. Setting the Supervisor Option NoDefActive will prevent the redefinition of active routines. The WHEN ? command will add a prefix of 't' to the name of any Opal routine which has been redefined in this way.

DEFINE * allows the redefinition of active Situations or routines linked to Situations. If DEFINE * is used then the new code will be used on the next evaluation for Situations, or Entry for other routines. Until the new code has been loaded, a WHEN ? will show a prefix of 'a' before the name of any routine where the code will be replaced. This command allows the code of an active event based SITU/ODTS to be replaced without having to stop and start the routines which could cause some events to be missed.

Since changing the code of an active slot may raise security concerns, the change will be logged. If the change is made from the ODT or a Supervisor window, a message is displayed for each affected slot. If the change is made from an Enter file, a message is logged to the Supervisor Log session. (TT WS lists this session number). A message is displayed after the Enter specifying how many slots were affected and also show the Log command which could be used to see the detail. When the new code is actually loaded (on the next Eval or Enter) an 'Situ reloaded' or "ODTS reloaded" message is displayed, indicating that either new Situation or new ODTSequence code has been loaded.

If site policy determines that changes to certain slots should not be permitted then the Situation ODTS and WHEN command should all be locked using the TT FOR command.

In order that Enter files can be created with DEFINE * instead of DEFINE +, a optional * can be used with the SAVE command. TT SAVE * DEF <pattern> or TT SAVE * SCHEDULE AS will create a file which has DEFINE * instead of DEFINE +

Version

The Version clause allows a numeric version to be stored with this Opal program. The version is displayed by the DEF ? command and can be checked in an opal program using MYSELF(CODEVERSION), not to be confused with MYSELF(OPALVERSION) which returns the version of the Opals compiler used to create the Opal.

Set/Reset

The set or Reset clause is used to set or reset any of the following three options for this Opal.

LIST

Causes or prevents a listing of the compiled opal to be printed

CODE

Causes or prevents a listing of the 'CODE' for this Opal to be printed. This is generally only used by Metalogic staff when debugging problems.

COMPRESS

This option is related to the SUPERVISOR Option called COMPRESSDEFINE. If this option is set (SO+COMPRESSDEFINE) then SUPERVISOR will use Metalogic's MZIP technology to compress Opal DEFINE sources as they are written into the SCHEDULE file. Although there is a CPU overhead associated with deflation (DEFINE +) and inflation (DEFINE ? and SAVE DEFINE), most DEFINE sources can compress to 20-25% of the original size or better for larger Opals.

Since the SO COMPRESSDEFINE setting applies to all DEFINE + or ENTER

actions, it is possible to override the default action by using the SET or RESET COMPRESS directive with an individual DEFINE:

```
DEFINE + ODTSEQUENCE TEST(MX) SET COMPRESS:
DEFINE + ODTSEQUENCE TEST2(PER) RESET COMPRESS:
```

Note that SUPERVISOR does not compress Opal sources that are less than 2 segments (360 chars) in length.

With the recent increase in the size of OPAL scripts that can be stored in the SCHEDULE from 255 to 2184 segments (see SUPERVISOR DNote 541.74) then even larger scripts can be stored if they are compressed with MZIP.

Program Types

MEMO

MEMOs are very simple program. A MEMO is not an OPAL program, and is used to write on-line help by associating documentation with an <OPAL id> using the DEFINE command, e.g.

```
DEF + MEMO DB_NOTES:
  THE FOLLOWING OPAL ROUTINES ARE AVAILABLE FOR DATABASE CONTROL.
  TT DO DB_CHECK MIX <DATABASE NAME>
    DISPLAYS ALL TASKS USING THE DATABASE NAME SPECIFIED
  TT DO DB_CHECK STATUS <DATABASE NAME>
    DISPLAYS THE STATUS OF THE SPECIFIED DATABASE
  TT DO DB_CLOSE <TYPE> <DATABASE NAME>
    SENDS AN SM CLOSE PROGRAM COMMAND TO COMS FOR EACH TASK OF
    THE TYPE SPECIFIED, WHICH ARE LINKED TO THE DATABASE.
    <TYPE> IS "INQ" FOR OPEN INQUIRY,
            "UPDATE" OPEN UPDATE,
            OR "ALL" FOR ALL PROGRAMS WITH THE DATABASE OPEN.
  FOR EXAMPLE:
    TT DO DB_CHECK MIX TOYSDBAUD
```

SITUATION

A SITUATION or SITU (which we will write SITUation to reflect the two alternatives) is used to find objects, such as a waiting mix entry. In OPAL, a SITUation is syntactically a Boolean expression, e.g.

```
DENSITY=BPI6250 OR SERIALNO<99
```

ODTSEQUENCE

An ODTSEQUENCE or ODTs (henceforth ODTSequence) is primarily used to group together a sequence of system commands that are understood to be one logical operational function. For example, some installations alter attributes of system queues at fixed times of the day, an operation that may require many individual system commands

The syntax of an ODTSequence is a sequence of statements, separated by semi-

colons, which describe an action to be taken upon the object (normally given by the Situation). Typical action might be to simulate an ODT input (ODT) or write a message on the ODT (SHOW).

Example

```
SHOW("Security Violation at ",SOURCESTATION);  
ODT("RWMT",UNITNO);
```

The following example will run JAMPACK compression against a pack designated by unit number:

```
DEFINE + ODTSEQUENCE EX_QUICKSQUASH(PER) :  
    DISPLAY(FAMILYNAME," # ",FAMILYINDEX," IS CHECKERBOARDED");  
    IF RUNNING("METALOGIC/JAMPACK") OR RUNNING("*RESERVEDISK")  
    THEN  
        DISPLAY("JAM IN PROGRESS: NOT DONE ON ",LABEL)  
    ELSE  
        ODT("TT JAM ",FAMILYNAME," # ",FAMILYINDEX," < ",  
            STRING(DU(FAMILYNAME,FAMILYINDEX) * .6,*));
```

COMMAND

COMMAND is a variant of ODTSEQUENCE and has the same syntactical implementation as that ODTSEQUENCE.

A COMMAND program allows the OPAL programmer to write ODTSEQUENCES under a program name that can be executed from a SUPERVISOR window without prefixing by 'TT DO'. One advantage of this is that a COMMAND program can be used to replace an Unisys operator command.

For example:

```
DEFINE + COMMAND C :  
SHOW(KEYIN("C FULL"))
```

The above simple COMMAND program allows the the usual Unisys 'C' command to be superseded by the SUPERVISOR equivalent; when C is entered from a SUPERVISOR window, *all* system COMPLETED entries will be shown since a 'C FULL' is processed instead. Note that any 17 character identifier can be assigned to a COMMAND program not just Unisys commands allowing specialised sequences to be assigned a shortcut.

Mix number related commands e.g. 1234Y may also be replaced by a COMMAND equivalent.

So:

```
DEFINE + COMMAND Y(MX) :  
$Y:= KEYIN(#(MIXNO,"Y"));  
$Y:= &#(/,KEYIN(#(MIXNO,"TI")));  
$Y:= &#(/,KEYIN(#(MIXNO,"FI")));  
SHOW($Y);
```

With Y defined as a COMMAND program, any <mixno>Y command issued from a SUPERVISOR window would give a more detailed status response including time and file information.

The file OPALS/SUPERVISOR/GETTINGSTARTED contains a much more comprehensive example of a Y command.

Once a COMMAND program has been compiled, it will immediately be available for use from any SUPERVISOR window. COMMAND programs can also be executed by the COM command.

Example:

```
TT COM C
```

DISPLAY

A DISPLAY or DISP (henceforth DISPlay) is a program that brings the powerful syntax of OPAL to bear on the problem of writing reports to the system console, REMOTESPO, or line printer. It is used to program output for new interrogation messages, and when used with SITUations, can allow a more flexible ADM facility.

Generally, a DISPlay shares a syntax similar to ADM EVENT PRINTLABEL but with fully-featured expression handling, simple but powerful formatting, and the ability to precisely control the terminal.

Example:

```
TDPAGE (2) , "#[SUB]D=", DENSITY 6
```

However, DISPlays can be quite complex. The following simulates the typical response one would get for a PER ODT command.

```
DEFINE + DISPLAY EX_PMT (PER) :
  UNITNO 3,
  (IF NOT AVAILABLETOGROUP THEN
    " NOT AVAILABLE TO GROUP"
  ELSE
    ((IF WRITEENABLED THEN "*" ELSE " ") &
    KIND&(IF VOLUMED THEN "\" ELSE " ") &
    " [& SERIALNO &"] "&
    (IF INUSE THEN
      IF MIX=0 THEN "(MCP)" ELSE "(&STRING(MIX,4)&)"
    ELSE
      "" ) &
    DROP(DENSITY,3) &
    (IF SCRATCH THEN " S C R A T C H "
    ELSE
      " "&"#"&STRING(REEL,*) & " "&STRING(CYCLE,*) &
      " : "&STRING(VERSION,*) & " "&TITLE) ))
```

OPAL Program Contexts

<context> is used to tell the OPAL compiler the <object class> of the program being DEFINEd. All OPAL programs have an object class, which is the domain on which the information available (ATTRIBUTES) is founded. A list of currently available OPAL contexts may be viewed, at any time, by entering TT HELP CONTEXT.

The SYSTEM context is a global, default context that is always available and so does not have to be explicitly declared in a DEFINE command. If no other context is given, SYSTEM is assumed. All OPAL programs of any context can access SYSTEM ATTRIBUTES in any context, but other ATTRIBUTES may only be accessed from a matching context.

For ODTS or DISP, <context> may **only** contain a major object class; no subtypes etc. are allowed. These are reserved for SITUATIONS only.

A <context> may **not** be specified with a MEMO.

LOG-based contexts

If used with WHEN, all LOG context OPALs are EVENT-based, feeding from the MCP entrypoint REPORT_LOG_ENTRIES. Log context WHENs cannot use the DELAY modifier nor can TIME be specified in a SITUATION.

If used with EVAL, Supervisor reads the current SUMLOG for all occurrences of the specified log record. The extent of log scanning can be controlled by start and end time/date, specific SUMLOG title or log number.

Context	Sub-Contexts	Notes
DATABASE	OPEN, CLOSE	Programmatic database open & close (Major Type 1, Minor types 19,20)
DMS	START, STOP	Database freeze and resume (Major type 1, Minor types 21 and 22)
EI		Establish Identity (Major type 0, Minor type 1)
FILECLOSE	CLOSE, PLI	Programmatic file close records (Major type 1, Minor types 6,10)
FILESTATUS	CREATIONCOPY, CREATION, REMOVAL, TITLE, SECURITY, COPY, DESTROY	File status change log records (Major type 16, Minor types 1,2,3,4,5,6,7,8)
JOBREJECT		Job rejected records (Major type 1, Minor type 7)

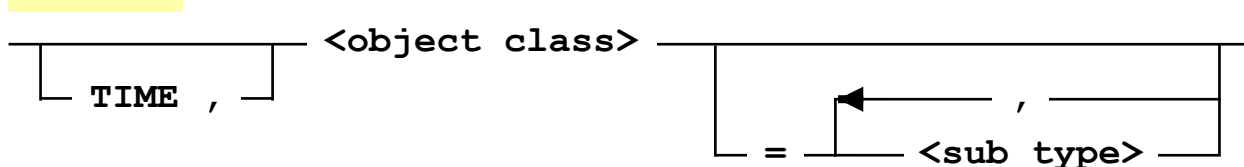
Context	Sub-Contexts	Notes
LOG	Optional Major, Minor	All log records filtered by Major/Minor types
LOGBOJ	BOJ, BOT	Log BOJ and BOT records (Major type 1, Minor types 1,3)
LOGEOJ	EOJ, EOT	Log EOJ and EOT records (Major type 1, Minor types 2,4)
LOGOFF		All MCS-generated LOGOFF records (Major type 4, Minor type 2)
LOGON		All MCS-generated LOGON records (Major type 4, Minor type 1)
LOGPS	CREATED, COMPLETION, STARTED, PRINTED	Job or task print request generation (Major type 1, Minor types 11,12,13,14)
MCSSECURITY		MCS Security Violation entries (Major type 4, Minor type 6)
MESSAGE		System/program messages (Major type 14)
OPERATOR	SETSTATUS, CONTROLLER, PS, PRIMITIVE	All Operator command types (Major Type 6, Minor types 3,7,8,12)
SECURITY		Security violation entry (Major type 6, Minor types 4)
USER		USERDATA change entries (Major type 6, Minor type 9)
VOLUME	ONLINE, OFFLINE, TAPEPG, TAPENew TAPEEXPIRY, TAPEUSE, TAPEHOLD	Log volume entries for tapes, packs, CDs (Major type 15, Minor types 1,2,34,5,12,13)
WINDOW	REMOTE, MCS, DIRECT	Log entries for COMS windows

Contexts in SITUations

<context> for ODTs or DISP consists of the identifier corresponding to the object

class. For SITUations, it has the following expanded syntax:

<context>



<object class> is the same as for the other <Program type>s. The list of <subtype>s is used to further restrict the objects that the SITUation should search. TT HELP CONTEXT lists the available <subtype>s, if any, after each <object class>. Most <subtype>s are identifiers, but integers may be used when the object class is LOG or NAPLOG.

When using the LOG <object class> the integers correspond to the major and minor LOGTYPES that should be returned. This is used mostly in cases where a particular log record does not have a corresponding <object class>. The first number is interpreted as the major LOGTYPE. If no further numbers appears, all minor types will be returned. Any subsequent numbers specify what minor types to return.

For example, a SITUation to detect LOG Major type 15 (Volume Status entry) could be:

```
DEFINE + SITUATION LOG_VOL (LOG=15) :  
    TRUE
```

would capture all minor types 1 through 5,12 and 13.

If only minor types 3 (Tape volume purged) and 5 (Tape volume NEWFILE) were required:

```
DEFINE + SITUATION LOG_VOL (LOG=15 , 3 , 5) :  
    TRUE
```

The NAPLOG object class can use a numeric <subtype> to implement a mechanism for filtering out certain URCs (Unique Reason Code), similar to NOAM, and uses a configuration variable to hold a number list.

For example:

```
DEFINE + SITUATION NAP_MONITOR (NAPLOG=3) :  
    TRUE
```

In this case, SUPERVISOR will search for a configuration variable named NAPFILTER_3 that could be set up using the INSTALL utility as follows:

```
U META/INSTALL NAPFILTER_3=3721,4567,1000-1040,890
```

Multiple filters can be established by changing the number suffix. When a NAPLOG WHEN, with a filter assignment, is invoked the NAPHANDLER task will check the validity of filter values and display a diagnostic message. If any errors are detected, an appropriate error is displayed and the filter discarded.

If the filter check is successful, then NAPHANDLER will instigate a call to an entrypoint in the NAPSUPPORT library called ALTER_NAPLOG_FILTER, This

inhibits the forwarding to NAPHANDLER of all NAPLOG entries whose URC code appears in the filter.

This mechanism offers an excellent way of ignoring common and unwanted URCs that would otherwise consume SUPERVISOR resources. Although SUPERVISOR will allow multiple NAPLOG filters to be active, the most efficient environment is to use just one active NAPLOG WHEN with its own extensive filter.

The USER `<object class>` can be used to scan the USERDATAFILE.

For example to scan for privileged usercodes:

```
DEFINE + SITUATION SCAN_FOR_PU (USER:PU)
    TRUE
```

A saved USERDATAFILE can be scanned in an EV command by using the TITLE modifier.

Example

```
TT EV (USER:PU) [TITLE=SAVED/SYSTEM/USERDATAFILE ON SV]
```

A TIME qualifier denotes that if a SITU is executed using a WHEN command, it should not be evaluated by attaching it to the Event corresponding to the `<object class>`, but instead should be evaluated periodically with an interval given by the DELAY (which defaults to 60 seconds).

The '-' qualifier is currently only valid for the PER object class, and mimics the usage in the PER ODT command, i.e. if the '-' is present, all units will be found, if it is absent, only READY units will be found.

For example:

```
EV (PER=MT- : TRUE)
```

will return all tape units known to the system. If the '-' is omitted; only visible units with tapes actually mounted and ready will be returned.

Evaluation of SITUations

SITUations can be invoked in two ways: by event and by evaluation. SITUations can be evaluated using the [Evaluate command](#). When a SYSTEM SITUation is evaluated, the Boolean expression is computed and a simple TRUE or FALSE is returned. A SITUation of any other object class gathers all the currently available objects of the class, further restricted by any subtype. If, for example, the context is (MX=WAITING), only the waiting tasks will be scanned. If a simple (MX) is used, all WAITING, ACTIVE, and SCHEDULED entries are scanned. For each task in the context, the Boolean expression is evaluated. The SITUation then returns either FALSE or TRUE along with a list of entities that it found. In general, if the SITU's Boolean expression returns TRUE, it will also return a list of the objects, which satisfy the SITUation.

A SITUation can be invoked by an [Evaluate command](#), or a [WHEN command](#). Evaluate gives the result of the SITUation and passes it one time only to the linked

ODTS or DISP, then it stops. WHEN evaluates the SITUation multiple times.

For the SYSTEM context, or if the <situ context> has the TIME directive, the SITU is evaluated periodically and, whenever the SITU returns TRUE, the linked ODTSequence or DISPlay is executed. The SITU will not be re-evaluated until the next delay period has elapsed and will continue in this cycle until explicitly stopped. Using the ONCE variant instead of the WHEN command, this cyclical behaviour does not occur since the SITU will be terminated after the first TRUE evaluation.

Certain SITUation contexts (e.g. COMPLETED, MESSAGE, NAPLOG) cannot be invoked by EVALuate; they can only be linked to an ODTSequence or DISPlay by a WHEN or ONCE command. These are passed event messages supplied by SUPERVISOR whenever a change occurs in the object class being monitored. Such SITUations can also never appear in a ONCE or WHEN which has a DELAY for evaluation.

A SITUation with a USER context can be invoked by an EVAL command to scan entries in an active or off-line USERDATAFILE. If a USER context is associated with the WHEN or ONCE command, the SITU will receive notifications of programmatic changes to the USERDATAFILE (SUMLOG records for Major Type 6, Minor Type 9)

SITUations of other object classes can be evaluated either by Event or TIME as specified in the DEFINE context.

In the special case of an EVAL of the USER context, SUPERVISOR will read the current *SYSTEM/USERDATAFILE and not the SUMLOG.

The generic LOG context can take an optional MAJOR and MINOR type:

```
DEFINE + SITUATION LOGCHK (LOG=21) :  
DEFINE + SITUATION LOGMAJ (LOG=12, 3) :
```

Non-LOG contexts

The following SITUATION contexts are not SUMLOG based.

The table below shows the permissions using WHEN and EVAL with non-SUMLOG OPALs. Note that not all sub-context synonyms have been included.

Context	Sub-Contexts	Notes	When	Eval	Time
AFTER	Sunday, Monday, Tuesday, Wednesday, Thursday, Friday, Saturday, Coldstart, Today	Scans the after specifications in the schedule file	No	Yes	No

Context	Sub-Contexts	Notes	When	Eval	Time
COMPLETED		Receives job/task EOT and EOJ. LOGEOJ is preferred.	Yes	No	No
CUSTOM		User definable context	YES	YES	NO
DBACCESS		Allows monitoring of DBACCESS Log Entries (Major type 1 Minor type 35).	YES	YES	NO
DEFINE	SITU, ODTS, DISP, MEMO, COMMAND	Scans the defined Opal programs	NO	Yes	NO
HTTP		Receives HTTP input via Port 8080	YES	NO	
JOBQUEUE	Integer	Analysis of job entries in SQ	No	Yes	No
METALOG	SUP (SUPERVISOR) ,TRIM,MAIL (MAILLIB)	Scans the select Metalogic Log file	No	Yes	No
MX	ACTIVE, LIBS, BOJ, SCHEDULED, DBS, WAITING, GOING, PRIORITY, BOT	All MX-based status changes	Yes	Yes	Yes
NAPLOG	Optional filter class (integer)	Unisys NAP log events	Yes	No	No
PD		Allows scanning of file directories	No	Yes	No
PER	MT, LP, DK, SC, VC CD, TSP, HY, CR CP, DC, PK, UD HC, FP, SPC, ARP, IP, NP, VSID, LBP	All PER-based status changes	Yes	Yes	Yes
PRINTS		Evaluates PS SHOW entries	No	Yes	No
SESSIONS		Returns data on sessions	No	Yes	No

Context	Sub-Contexts	Notes	When	Eval	Time
SHOWOPEN	DK, PK, CD	Returns open files on disk, pack or CDs	No	Yes	No
SL		Scans the MCP Sled functions	No	Yes	No
SMTP		Used to receive email messages	Yes		
STATIONS		Reports on MCP datacom stations	No	Yes	No
SYSTEM		System-based data	Yes	Yes	Implicit
TAPEDB	PENDING, NAME, FAMILY, POOL, TS, SCRATCH, LOCATION	WHEN gets tape notice from Supervisor. EVAL scans TRIM database	Yes	Yes	No
TAPELABEL		WHEN gets raw tape labels from MCP. EVAL scans TRIM database	Yes	Yes	No
USER		WHEN receives raw log entries. EVAL reads USERDATAFILE	Yes	Yes	No
VDBS		Scans active databases	No	Yes	No
VL	DISK, PACK, TAPE CD, CDIMAGE, FARM	Scans the system Volume Library. Cataloging systems only.	No	Yes	No
WHEN		Scans the active When slots	Yes	Yes	Yes

Context Notes

Only four contexts can be time-based (i.e. polling); these are PER, MX, SYSTEM and WHEN. Both PER and MX are event-based by default and require the keyword TIME to force the SITUATION to poll. The SYSTEM context is never event-based and implicitly uses TIME. Note that PER and MX, with TIME, do **not** receive event changes from the MCP.

By default, all TIME based SITUATIONS have a delay of 1 minute. This can be

modified using the DELAY keyword when running a WHEN e.g.

```
TT WHEN PER DO PER DELAY 45 SECONDS
TT WHEN MIXCHK DO MIXCHK DELAY 10
```

TAPEDB WHENs are a special case; TRIM generates these notices after tape events (the raw MCP tape notices appear as TAPELABEL events) have been applied to the database and passes them to SUPERVISOR for processing.

HELP ATT =:<context> will list all attributes for a context

Some of the non log contexts are described in more detail below.

AFTER Context

This context permits access to SUPERVISOR's AFTER schedule and has many attributes that map information related to an individual activity. This access is only available to the EVAL command.

The AFTER context allows the activity schedule to be filtered by the day of the week (SUNDAY=>SATURDAY) or by the special keywords TODAY and COLDSTART. For example:

```
EVAL (AFTER=SUNDAY , TUESDAY : TRUE)
```

The above example would only examine activities that normally appear in the schedule for ON SUNDAY and ON TUESDAY.

Various attributes allow the caller to filter by many different characteristics. For example, boolean attributes such as ONHALTLOAD, ONRESTART, ONSUNDAY etc. allow the user to filter individual activities by a variety of different conditions. String attributes such as ACTIVITY, FORTEXT, ADDTEXT and DELETETEXT allow information about the schedule entry text content to be retrieved. Please see HELP ATTR =:AFTER for a complete list of the attributes in this context.

Similar to the syntax used for LOG-based EVALS, it is also possible to control access to schedule records by using time of day and/or date ranges filters. The syntax is slightly different from that used by LOG and both time and date ranges can be used.

For example:

```
EV (AFTER:TRUE) [@1000-1300 20/2/08] DO (SHOW(ADDTEXT))
EV AFSITU [@21/02/08-23/02/08] DO AFODTS
```

Using a time range without a date or date range will cause the general schedule to be filtered by time only. In the event a date is specified, the SCHEDULEDAY and SCHEDULEDATE attributes will be automatically set to the appropriate date.

AFTER context programs may also be executed standalone with the DO command. To retrieve information for a unique record in the schedule, the DO command must be given the schedule day or specific julian ON DATE, the minute of day and the activity number. The Opal KEY attribute provides this access but its use is not intended for general use.

This interface has been provided to support functionality in the new Metalogic Web interface, which operates with a variety of internet browsers, and offers direct access into the SUPERVISOR schedule.

CUSTOM Context

A Custom Context is a Context which is implemented by a customized Library external to Supervisor. There may be many Custom Contexts, each Context being selected by an Interface in an SL Library.

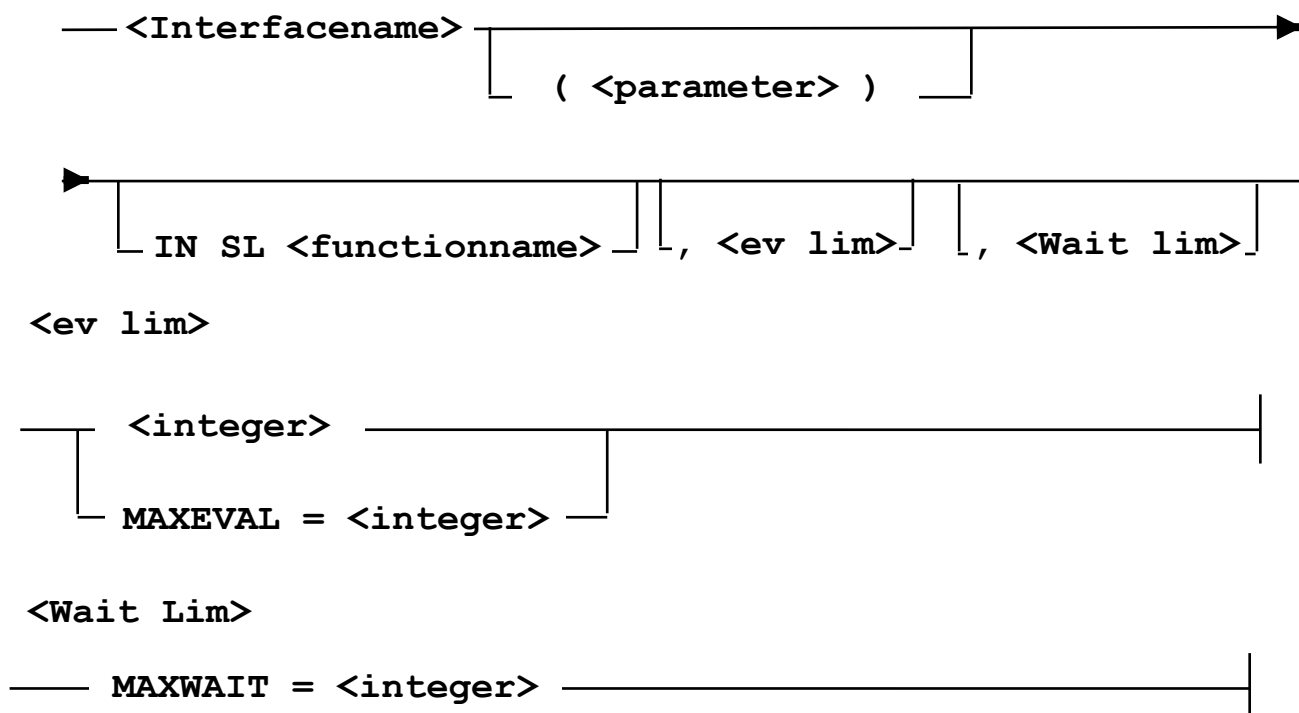
For example, a Custom Context WHEN might wait for some event which is triggered from another computer. A Custom Context EVAL or DO might process a proprietary file or database and return selected records.

A Custom Context SITUATION, ODTS or DISPLAY is defined using the DEFINE command,

```
DEFINE + SITU TESTSITU (CUSTOM) : TRUE
DEFINE + ODTS TESTODTS (CUSTOM) : Display (ToString) ;
DEFINE + DISPLAY TESTDISP (CUSTOM) : ToString;
```

A Custom Specification is used to define the INTERFACENAME and the Library FUNCTIONNAME of a Custom Context Library, as well as a string parameter which may be used for initialization.

Custom Specification



The <interfacename> is used to connect to a Connection Library referenced by the

SL <functionname>.

The <parameter> may contain any text, except a <right parenthesis>. If a <right parenthesis> is required, the item of which it is part must be enclosed in double quotes. For example FILE="(A)B ON C".

The only required item is <interfacename>. The (<parameter>) may be omitted. If the IN SL <functionname> is omitted, then the <functionname> defaults to the reserved functionname, METACONTEXTS.

The Custom Specification is provided when the WHEN, EVAL or DO is initiated.

```
WHEN TESTSITU [TEST IN SL TESTCONTEXT] DO TESTODTS <text>
WHEN TESTSITU [TEST IN SL TESTCONTEXT] DISPLAY TESTDISP
<text>
EVAL TESTSITU [TEST IN SL TESTCONTEXT] DO TESTODTS <text>
DO TESTODTS [TEST IN SL TESTCONTEXT] <text>
DISPLAY TESTDISP [TEST IN SL TESTCONTEXT] <text>
```

The <text> is passed to the Custom Context Library during the setup of the slot.

There are currently 7 predefined attributes in the Custom Context.

TOSTRING	Implemented by the Custom Context, it returns the current object as text.
LASTERRORTTEXT	If an attribute which was accessed was not valid, this attribute may contain the reason.
HELP	Binds to an attribute called HELP in the Custom Context Library and by convention, returns information about the Custom Context.
ATTLIST	Binds to an attribute called ATTLIST in the Custom Context Library and by convention, returns a list of attributes either comma separated, HTML or JSON, depending on an optional parameter.
FUNCTIONNAME	The SL FUNCTIONNAME for the Context Library.
INTERFACENAME	The INTERFACENAME for the Context.
LIBPARAMETER	The parameter passed with the Custom Specification.

The MYSELF(INPUTPARAM) function returns the WHEN,EVAL,DO or DISPLAY parameter. The MYSELF(USER) function returns the associated UserCode.

The capability to reference dynamically defined attributes in Custom Context OPALs will be released later.

The DEBUG + 34 Option may be used to assist in debugging Custom Contexts, with the messages being stored in the SUPERVISOR Log.

The Custom Context Library must implement a Connection Library which exports the

following entrypoints,

```
Boolean PROCEDURE SetUp (SlotNo, WhenType, UserName ,  
                          Situation, ODTSequence ,  
                          LibParameter, WhenParameter ,  
                          MaxObjects, Options, Reason) ;
```

MaxObjects corresponds to the Eval Limit of an EVAL or the Do Limit of a standalone DO.

If the SetUp function returns TRUE, then the Custom Context Library may start inserting objects into the WhenEvQ.

SetUp MUST NOT wait on anything as this will hang SUPERVISOR.

```
String PROCEDURE ToString(A) ;
```

Returns a string text representation of the current object (A).

```
String PROCEDURE Reference Array GetStringAtt[0:MaxAttributes] (A, ID, AttInfo, P1, P2, P3, P4, P5) ;
```

Called to return a string attribute value. A is the current Object, originally provided by the Custom Context. ID is the uppercase name of the attribute. P1 through P5 are parameters to the attribute or EMPTY.

```
Integer PROCEDURE BindStringAtt (ID, AttInfo, Semantics, Reason) ;
```

Called to return the Index in GetStringAtt of the Procedure which implements the code for the string attribute called ID. If the Index < 0 then the Reason is logged to the SUPERVISOR Log and the attribute returns EMPTY.

```
PROCEDURE LogOff (WhenType, Why) ;
```

Called to notify the Custom Context Library that its all over and no further messages should be inserted into the WhenEvQ.

The Custom Context Library generates Objects (DCALGOL Messages) and inserts them in the WhenEvQ of the OPAL Slot. The layout of the Object is defined by the Custom Context Library, with the restriction that it must be more than 1 WORD.

If SUPERVISOR receives an Object which is 1 WORD in size, then it indicates the End of Objects. The convention is that if the 1 WORD IS 0, the OPAL finished successfully, otherwise there was an error and the contents of the 1 WORD are logged to the SUPERVISOR Log.

Example TEST Context

There is an example Custom Context Library called

```
*TESTCONTEXT/LIBRARY/NEWTAPE
```

Example FTP Context

Another example of a SUPERVISOR Custom Context is FTPCONTEXT which allows an OPAL script to send commands and receive replies from an FTP Server.

An FTP context OPAL ODTs and SITU are defined in the example below,

```
DEF + SITU IC(CUSTOM) :  
    TRUE  
DEF + ODTs IC(CUSTOM) :  
    Show(ToString) ;
```

In this example, the only attribute referenced is the TOSTRING Custom Context attribute which returns a description of the current FTP response.

A Directory Scan can be initiated with an EVAL.

```
EVAL IC [DIRECTORY IN SL FTPCONTEXT]  
    DO IC ftp://user:pass@192.168.1.31/DO/
```

The usercode and password may need to be supplied, depending on the FTP Server. If the FTP Server allows anonymous access, the usercode should be anonymous. The FTP context puts in the user@hostname as the password if a password is required.

The port defaults to 21, but may be specified after the hostname, for example 192.168.1.1:1234

If the ftp url ends with a /, then by default a list of the filenames is returned using the NLST command. Each file name is returned in a separate response with a message type of FILE.

If the ftp url does not end with a /, then the STAT verb is used which returns either the details of a single file, or all the files in the directory, in a single response, with logical lines separated by a CRLF.

The FTPCONTEXT only supports IMAGE Mode, and all responses are returned in EBCDIC. Data provided to FTP interface attributes are treated as binary.

An EVAL will terminate at the end of the directory scan. An FTP dialog can be scripted using a DO.

The initial command used by the DIRECTORY interface may be specified as a Library Parameter, and may be either LIST, NLST or STAT, as in this example:

```
EVAL IC [DIRECTORY(STAT) IN SL FTPCONTEXT] ...
```

A LIST command returns a formatted directory listing suitable for people to read.

An NLST command returns only the names of the files or directories, suitable for a computer to read.

The STAT command, when used with a directory returns a formatted directory listing, over the control port.

A Directory Scan can be initiated with a DO in which case the FTPCONTEXT is in scripting mode. As FTP responses are received from the FTP Server, the OPAL script can issue further FTP commands.

```
DO IC[DIRECTORY IN SL FTPCONTEXT]
DO IC ftp://user:pass@192.168.1.31/DO/
```

As for an EVAL, if a directory is specified (a trailing /) then an NLST command is executed, otherwise a STAT command.

A DO will not terminate at the end of the directory scan. The FTPAPI sends NOOP commands to the FTP Server while it awaits further FTP commands. The OPAL receives the response to the NOOP commands, until the slot terminates, either by using the CLOSE method, or by DO-#<slotid>.

The initial url is used to navigate to some level within the directory structure. The names of files give in a path parameter should therefore be relative to the initial directory.

The current path is requested during the LOGON sequence, and whenever the SETPATH request is made. The command is PWD (Print Working Directory), and the response is received by the script.

These methods are provided for a dialog with an FTP Server. A method is an attribute with parameters, which returns a result. These methods queue requests to the FTPCONTEXT. All methods return a string value, which is either ERROR:<text> to indicate an error, or <integer> which is the REQUESTID, and may be used to identify responses returned by the FTP Server.

- CLOSE()
Sends an end of objects indication to the slot, so that the LASTEVAL, if defined, should occur.
- DELETEFILE(Path)
Requests the deletion of the file with the specified path.
- FTPHELP(Key)
If Key is empty, general help is request from the FTP Server, otherwise specific help for the Key.
- FTPSITE(Command)
Sends a site specific command to the FTP Server.
- FTPSTATUS(Path)
If the path is empty, the current path is used, otherwise the file information for the specified path is requested.
- GETBYTES(Path)
Requests the file specified by path as a byte string. This is limited by the Algol string size of 65535 bytes (MCP 54).
- GETDIRECTORY(Path,AsFile)
Requests the directory listing in a file if an AsFile title is specified, otherwise as a string.

- **GETFILE(Path,AsFile)**
Requests the file specified by the path. If an AsFile title is not specified, then a unique MCP TITLE is created.
- **MAKEDIRECTORY(Path)**
Requests that the specified directory, with reference to the current path, be created.
- **PUTBYTES(Data,AsPath)**
Stores the Data with the given AsPath. The Data is only limited by the size of an OPAL string, which is currently 2 MB.
- **PUTFILE(Title,AsPath)**
Stores the file specified by the Title with the AsPath. Only Byte Stream Files are accepted.
- **REMOVEDIRECTORY(Path)**
Requests removal of the specified directory, with reference to current path.
- **RENAMEFILE(FromPath,ToPath)**
Requests the renaming of the file FromPath to ToPath.
- **SETPATH(Path)**
Changes the current path to an absolute or relative pathname.

When a method is invoked, one or more FTP Commands are sent to the FTP Server. These commands make up a Request, which is bounded by a BOT and an EOT or ERROR message.

Requests are stored and executed in the order they are received. A new Request may be queued at any time within the OPAL script.

These are the attributes which are available each time the OPAL script is entered, because of a response from the FTP Server.

- **COMMANDNAME**
The actual FTP command name which generated the response, in UPPER case.
- **COPIEDDATA**
Returns the string TRUE if there is data associated with this request, for example after a GETBYTES request. Otherwise it returns EMPTY.
- **COPIEDFILE**
Returns the string TRUE if a file was created to contain the data returned from the FTP Server, for example after a GETFILE request. Otherwise it returns EMPTY.
- **COPYDATA**
If COPIEDDATA, then this attribute returns the data from the FTP server.

- FILENO
Returns the File Number of the file in the original request if a / directory NLST request was performed.
- FILESTATE
Returns the Port FILESTATE of the Data Transfer PortFile (DTP).
- LASTFILE
Returns the string TRUE if this is the last file in an NLST of a directory request.
- MSGSIZE
Returns the length in bytes of the response message.
- MORETEXT
If the response is longer than a DCALGOL MESSAGE, it is segmented and this attribute returns the string TRUE, to indicate that more text follows. Apart from the MSGSIZE and MSGTEXT attributes, all other attributes for each segment are identical. If the response is not segmented, the attribute returns EMPTY.
- MSGTYPE
Returns the name of the type of this response message.
 BOT - Indicates the start of a request.
 CONNECTED - The FTP Server has accepted the logon.
 REPLY - An FTP command reply.
 EOT - Indicates the end of the request.
 ERROR - Indicates the end of the request, which failed due to an error.
 PROGRESS - A progress response giving the amount of data sent or received so far.
 FILE - For an NLST directory request, this is a File Name.
 USER - This is a message generated by a User application such as FTPCONTEXT.
- REPLYCODE
The numeric reply code as a string, for the associated FTP command.
- REQUESTID
The name of the original request. It is one of LOGON, GETDIRECTORY, GETFILE, PUTFILE, GETBYTES, PUTBYTES, SETPATH, DELETEFILE, REMOVEDIRECTORY, MAKEDIRECTORY, RENAMEFILE, HELP, SITE or STATUS. It is in upper case.
- REQUESTNO
The request number for this request, which corresponds to the number returned from the request method.
- TAG
A tag associated with the request.

If there is a collision with OPAL attributes, then all the FTP attributes above have a synonym of FTP_<attname>.

These custom attributes are also available,

- **TOSTRING**
Returns a string which gives a general description of the response from the FTP Server.
- **MYSELF(INPUTPARAM)**
This provides the original url provided with the DO or EVAL.
- **LIBPARAMETER**
This returns the library parameter provided with the DO or EVAL (Empty,NLST,STAT or LIST).

When storing or retrieving files from an FTP Server, the FTPAPI attempts to use MODE-Z, which provides for ZLib (Deflate) compression. Some FTP Servers such as FileZilla implement MODE-Z, however many do not. In this case, the FTPAPI will revert to MODE-I.

The SL FTPCONTEXT = *METALOGIC/SUPERVISOR/FTPCONTEXT requires the SL FTPAPI = *METALOGIC/COPYWRITE/FTPAPI and the SL TELNETAPI = *METALOGIC/COPYWRITE/TELNETAPI Libraries.

There is an example of a custom FTPCONTEXT Script called IC in, *EXAMPLES/SUPERVISOR/IC

This example scans the Unisys FTP Server for new ICs (Interim Corrections), which are then downloaded and unzipped and unwrapped to release the component files.

It uses the scripting features of the FTPCONTEXT so it is started as a DO, FOR USER/PASS

```
DO IC[DIRECTORY(STAT) IN SL FTPCONTEXT]
    ftp://anonymous@ftp.support.unisys.com/pub/aseries/
CONTAINER/054.1A
```

Example XML Context.

The XMLCONTEXT is a SUPERVISOR Custom Context which allows an OPAL script to process an XML Data File.

```
Define + ODTS RSS_Test(Custom):
    If (EVENTTYPE = "STARTELEMENT") AND (LOCALNAME="link") then
        #Link:=1;
    If (EVENTTYPE = "ENDELEMENT") AND (LOCALNAME="link") then
        #Link:=0;
    If (EVENTTYPE = "CHARACTERS") AND (#Link = 1) then
        Display(TEXT);
DO RSS_Test[XML IN SL XMLCONTEXT] (TEST)RSSTEST ON NUCLEUS
```

In this example, an RSS feed was downloaded into the file (TEST)RSSTEST using an HTTP(XHR) OPAL, and then processed using the XML OPAL RSS_Test, defined above.

The XML Context only supports a DO or an EVAL.

The Parameter to the XML DO or EVAL may be either a file which contains an XML Document, or an XML Document which begins with <?xml..., however in this case the XML Document is restricted to fewer than MAX_DCMESSAGE_SIZE bytes.

The XML Context interface is based on the SAX ContentHandler Interface.

The EVENTTYPE attribute specifies the type of Object, and it corresponds to a method of the SAX ContentHandler interface.

The EVENTTYPES and their associated attributes are,

"STARTDOCUMENT"	
"STARTELEMENT"	NamespaceURI,LocalName,QualifiedName,Prefix,Attributes
"ENDELEMENT"	NamespaceURI,LocalName,QualifiedName,Prefix
"CHARACTERS"	Text
"PROCESSINGINSTRUCTION"	Text

Non-SAX EVENTTYPES

"PARSEERROR"	Text
--------------	------

These attributes are defined,

EVENTTYPE	The type of object being processed by the OPAL code.
LEVEL	The hierarchical level of the element being processed. The Document is Level 0, and the Root Tag is Level 1.
NAMESPACEURI	The NameSpace URI either on the element, or inherited by the element, if specified.
LOCALNAME	The local name of the Tag, without any NameSpace Prefix.
QUALIFIEDNAME	The local name of the tag, with a NameSpace Prefix,if specified.
PREFIX	The NameSpace Prefix if specified.
ATTRIBUTES	The list of attributes and their values if specified on a Tag. It is returned in JSON Format, suitable for use with the Distribute(",","JSONObject) OPAL Function.
TEXT	For "CHARACTERS" it is the text content of the tag, including any WhiteSpace. For a "PROCESSINGINSTRUCTION" it is the content of the Processing Instruction. For a "PARSEERROR"it gives the reason.

DTD Entities are not decoded. The Translate(EntityDecode) OPAL Function may be used for intrinsic entities such as ampersand.

Example C74 Context.

The C74 CONTEXT is a SUPERVISOR Custom Context which allows an OPAL script to process a COBOL74 Data Definition.

```
DEFINE + ODTs C74 (CUSTOM) :  
  If (ELEMENTARY="TRUE" AND FILLER="FALSE") then  
    Display (LEVEL, , NAME, , PICTURE, , USAGE) ;  
DO C74 [C74 (TITLE="*SUPERVISOR/C74CONTEXT/TESTDATA ON DEV")  
      IN SL C74CONTEXT]
```

The Object for the C74CONTEXT is a Data Item Definition.

These attributes are defined:

```
BLANKWHENZERO, BYCONTENT, BYREFERENCE, COLUMNSPECIFIED,  
ELEMENTARY, FILLER, JUSTIFIED, LEADINGSIGN, LEVEL, NAME, OCCURS,  
OCCURSMAX, OCCURSMIN, OFFSET, PICTURE, REDEFINE, SCOPE, SEPARATESIGN,  
SIGN, SIZE, SYNCHRONIZE, USAGE, WITHLOWERBOUNDS.
```

The OFFSET attribute gives the offset in 4-Bit Digits.

The SIZE attribute gives the size in Bits.

HTTP Context

An event based context which receives http requests from browsers addressed to the domain of this host on port 8080. See OBI for detailed used of this context.

DO of an HTTP ODTs can be used to fetch resources from remote HTTP Servers.

It requires the SL XHR = *METALOGIC/XHR Library, which implements the XMLHttpRequest Object defined by the W3C at <http://www.w3.org/TR/XMLHttpRequest>.

The parameter to the HTTP ODTs defines the XMLHttpRequest, and has this syntax.

```
<method> <url> <user> <sync> [ <header>\<header...> ] ;<body>  
<method> ::= <text>  
<url> ::= "<http url>" | <http url>  
<sync> ::= SYNC | ASYNC | <empty>  
<user> ::= FOR <usercode>/<password> | <empty>  
<header> ::= <headerid> : <header value>  
<body> ::= <bytes>
```

Example:

```
GET http://192.168.1.31:80/main.htm?type=test FOR NFT/NFT sync  
[Content-type:plain/text\accept:gzip]
```

An HTTP ODTs which is initiated by a DO must have an associated USERCODE. The USERCODE can be associated by starting the ODTs with a FOR <usercode>/<password>, or by starting the ODTs from a SUPERVISOR Window (whether or not

it is logged in). This is the Initiator's USERCODE and it dictates where files which have been retrieved are stored and where files that are to be sent can be found.

The parameter to an HTTP ODTs DO may need to specify a <usercode>/<password> to access a remote system's resource, for example a web site that uses Basic Authentication.

The Metalogic HTTP Server provides an MCP based authentication called RU Authentication, which uses the MCP's UserDataFile RU mechanism.

If the parameter to an HTTP ODTs DO does not provide both a <usercode> and <password>, so either nothing or just a <usercode>, and the remote host supports RU Authentication then the <usercode> is checked against the RU tables in the UserDataFile for the IP address and Domain Name.

DEFINE Context

The DEFINE context enables access to the SUPERVISOR Opal program dictionary and detailed information about all DEFINES and MEMOs held in the SCHEDULE file. Opal scripts that use DEFINE may only be executed by the EVAL or DO command and no WHEN variant is permitted.

A complete list of the attributes can be seen:

HELP ATTR =:DEFINE

The DEFINE context may use subcontexts to filter the dictionary; these subcontexts are SITU,ODTS,DISP,COMMAND and MEMO. If one or more subcontexts are specified, only those DEFINES of the correct type will be returned.

These program types also map to the TYPE attribute.

EVAL (DEFINE=ODTS,DISP:NAME HDIS "A")

The above EVAL will examine all ODTSEQUENCE and DISPLAY programs in the dictionary additionally filtering them by using the NAME attribute to those whose id starts with 'A'. Note that, in the present implementation, the list of DEFINES returned is unsorted and reflects the order in which they were entered into the SCHEDULE dictionary.

Useful attributes include information about the program context (CONTEXT), the source of the DEFINE (SOURCE), whether it is currently in-use by a WHEN slot (INUSE, CALLED) and creation and access timestamp information.

When a DEFINE script is executed by a DO, SUPERVISOR expects a single level identifier that denotes the name of a DEFINE. All instances of that DEFINE will be returned and executed by the script and no filtering by program type is performed.

For example:

```
DO (DEFINE:SHOW(TYPE,,NAME 17,,CACHEDSLOTS) OBI_EXAMPLE
SITU OBI_EXAMPLE          0
ODTS OBI_EXAMPLE          2
DISP OBI_EXAMPLE          0
```

In the above case, OBI_EXAMPLE has three occurrences in the SCHEDULE and
OPAL related commands

the ODTS is currently active and cached by 2 WHEN slots. Note that SITU and MEMO program types can never be cached.

By default, all privileged users can view the entire DEFINE dictionary but non-privileged users may only view DEFINES locked under their own usercode or normal, unprotected DEFINES.

DBACCESS Context

Allows monitoring of DBACCESS Log Entries (Major type 1 Minor type 35).

See HELP ATT =:DBACCESS for details of available attributes

KEYS Context

Allows an OPAL script to process a KEYSFILE.

```
DEFINE + ODTS KEYS (CUSTOM) :  
    If Expired="TRUE" then  
        Show(Name, ,ExpiryDate) ;  
DO KEYS[KEYS IN SL KEYSCONTEXT] { <filetitle> }
```

If a <filetitle> is not provided, then *SYSTEM/KEYSFILE ON <hlfamily> is assumed.

Both EVAL and DO are supported for the Keys Context.

The KeysFile must have a FILEKIND of KEYSFILE.

The Object for the KEYSCONTEXT is a Key Record from a KEYSFILE.

The attributes defined include:

```
ALLOWREKEY, ALLOWRELICENSE, EXPIRED, EXPIRYDATE, HEXKEY,  
INSTALLTYPE, LICENSEID, NAME, PASSWORD, VERSION.
```

A Keys Item has been added to the Operating System Group in OBI. It can be used to produce a report on the SYSTEM/KEYSFILE.

The EXPLORE Action in a PD Report on a KEYSFILE produces a Keys Report using the selected Keys File.

PD Context

This context is the same as the implicit Directory context in FLEX. At this stage, PD context OPALs can only be invoked by a Do, Test, /, or Eval. The directory to be searched is placed after as a parameter.

Example

```
TT DO SAFEREM A/B ON C  
TT EVAL (PD:SEGMENTS > 1000) DISP (TITLE, ,SEGMENTS) SYMBOL/=
```

If no '/=' is present, Supervisor assumes that it is a single file, as in the MCP's PD ODT command. If the '/=' is used, a maximum of 255 objects will be returned. This restriction does not apply to users with a Flex licence who use a Flex option after the directory.

Example

```
TT / PDDISP (META)= ON SYSPK:Flat
```

Just as in the MCP PD ODT command, an unsigned integer can be placed after to restrict the depth of the search.

Example

```
TT / PDDISP (META)OBJECT/= ON SYSPK 3
```

Flex Inquiry Reports and ODTSequences are compatible with Supervisor and can be used directly. However, some of the intrinsic reports for :CAT and :CAT,ATT are badly formatted due to the differences in ODT and CANDE screen handling.

Clients with a FLEX Inquiry licence, are able to use the full power of FLEX from Supervisor, including wild cards, high speed searching, and unlimited returned objects.

MAIL Context

Mial is an event-only context that captures notices generated by the Metalogic MAILLIB library. These notices essentially reflect the information that appears in the MAILLIB log file but are much more detailed. The event notices are subdivided into a number of categories denoted by the EVENT and EVENTTYPE attributes.

These types are:

```
SND, RCV, ERR, MSG, FLT, QUE, ATT, INC
```

Available attributes are dictated by the various event types and can be checked by using the VALID function. For example, the SMTPFILENAME attribute is only valid for EVENTTYPE=QUE. Email-specific attributes such as TO, FROM, CC, FROM and MESSAGE SIZE are only relevant for notices whose EVENTTYPE is SND, QUE or RCV.

Example:

```
DEFINE + SITUATION MAILQ(MAIL) :  
    EVENTTYPE=QUE  
DEFINE + ODTSEQUENCE MAILQ(MAIL) :  
    RECORD[SNMP] ("Warning! EMAIL queued,  
Error=",ERRORNO," : ",TEXT) ;
```

In this simple case, the TEXT attribute returns the message entry that is written into the MAILLIB log file.

The WHEN is invoked by:

```
WHEN MAILQ DO MAILQ
```

Note that MAIL Opal scripts may only be used by the WHEN command. The METALOG context should be used to retrieve retrospective MAILLIB information but this is not as detailed as event-based info.

Each MAIL notice, regardless of EVENTTYPE, contains a subset of MAILLIB

configuration attributes that can be used in a MAIL script. Originally, these attributes were generally available in the SYSTEM ML attribute group but this set has now been renamed and each attribute name is now prefixed by 'MAIL'. For example the SYSTEM ML attribute 'POSTMASTER' is now called 'MAILPOSTMASTER' so any scripts that use these attributes will have to be recompiled with corrected names, though existing compiled code will continue to run.

If MAILLIB is terminated whilst any MAIL WHENs are active, SUPERVISOR will be forcibly delinked from the MAILLIB library but the WHENs will continue to remain active. When MAILLIB is restarted (by whatever means), MAILLIB will send a notice to SUPERVISOR to cause automatic re-linkage to the MAILLIB library to continue notice processing. If no WHENs are active, the request is ignored.

METALOG Context

This context allows retrieval of log entries from Metalogic's internal log files maintained by SUPERVISOR, MAILLIB and TRIM. Unlike SUMLOG-based scripts, METALOG Opals can only be used with the EVAL command as no event support is available.

The context has three subcontexts: SUPERVISOR, MAILLIB or TRIM. If no subcontext is given, SUPERVISOR is assumed by default; only one subcontext may be assigned. A small subset of attributes is available, similar to those available to the LOG context. The attributes can be viewed with the command `HELP ATTR =:METALOG`.

The implementation of METALOG log file browsing uses the same EVAL mechanism adopted by all LOG contexts. This allows start and end timestamps to be passed to the EVAL to filter the data returned. The EVENT attribute is particularly useful since this string value maps to the log entry category e.g. 'Err', 'Flt', 'Msg', 'Odt' etc.

For example:

```
EVAL (METALOG:EVENT="Err") DO (SHOW(LOGTEXT)) [12]
EVAL (METALOG=SUPERVISOR:TRUE) [@BACK 12 HRS FOR 20 MINS] DO META
```

However, it is not yet possible to use TITLE or LOGNO in the EVAL specification nor are some log-based MYSELF attributes (such as LASTLOGNO, LASTLOGTIME or LOGRECORDS) currently available.

All METALOG EVAL calls will now use the EVALREADER mechanism (as with LOG context EVALs). This means that there is effectively no limit to the number of log records scanned so very long time-range EVALs can be performed if the relevant log files are present.

If no time specification is provided then a mandatory maximum of 500 evaluated log records is applied.

RSTCONTEXT

The RSTCONTEXT is a SUPERVISOR Custom Context for processing the MCP RESIZE TRACE Diagnostic Information.

When the ODT Command DO + RST is entered, the MCP initializes its Resize Trace Tables. When a program does a RESIZE, diagnostic information is stored in the tables. The RSTCONTEXT provides access to this information.

The DO - RST ODT Command disables the RESIZE Trace and deallocates the Resize Trace Tables.

An RSTCONTEXT ODTs is a Custom Context ODTs.

Define + ODTs RST(CUSTOM) :

Display(ToString) ;

An RSTCONTEXT ODTs is initiated as a DO or an EVAL, and the Object returned is an entry from the RESIZE Trace Table.

DO <odts>[RST ({path}) IN SL RSTCONTEXT] {pattern}

The optional <path> may be used to specify the family substitution which will be used to find any codefile which does not have a family name. For Example, DO <odts>[RST(DISK = NUCLEUS ONLY) IN SL ...

The optional <pattern> may be used to filter the CodeFile of the caller of RESIZE. It behaves the same way as the EQW (WildCard Equals) Operator in OPAL.

These attributes are defined for the RSTCONTEXT:

ATTLIST

Returns a list of the Attributes in the RST Context.

CALLERCODE

Returns the CodeFileName that contains the code of the caller of the function that did the RESIZE.

CALLERCODETS

Returns the CREATIONTIMESTAMP of the CALLERCODE file.

CALLERLINE

If available, it returns the LINEINFO for the CALLERRCW.

CALLERRCW

Returns the RCW to the caller of the function that did the RESIZE.

HELP

Returns Attribute Information for the RST Context.

RESIZECODE

Returns the CodeFileName that contains the code that called the RESIZE function.

RESIZECODETS

Returns the CREATIONTIMESTAMP of the RESIZECODE.

RESIZECOUNT

The count of the times that the RESIZE function was called from the same code sequence.

RESIZELINE

If available, it returns the LINEINFO for the RESIZERCW.

RESIZERCW

Returns the RCW to the call on the RESIZE function.

TOTALRESIZES

The total number of calls on RESIZE since the last Halt/Load.

TOSTRING

Returns a summary of the RESIZE information.

The Resize Trace Tables do not contain the FAMILYNAME of the CodeFiles associated with a RESIZE, so the <path> and <pattern> may be used to locate a specific codefile.

SESSIONS Context

Information about the session include LSN, MCS number and name, Station name and log-on time.

Examples:

```
EVAL (SESSIONS:TRUE) DO (SHOW(SESSION 5,,STATIONNAME 40,,MCSNO))  
DO (SESSIONS:SHOW(SESSION,,LSN)) 240,255,256
```

The latter would show session information for sessions 240,255 and 256 assuming that they are valid.

In addition, SUPERVISOR shows meaningful station names for all of its own MCS sessions. Any WHEN, EVAL or DO slot that is active will include slot information e.g. SUPERVISOR/SLOT_3/WHEN would indicate a WHEN run in slot 3.

A complete list of SESSIONS attributes may be viewed by using the HELP ATTR=:SESSIONS command.

SHOWOPEN Context

The SHOWOPEN context allows the retrieval of information about all open, in-use files on a specified disk family and includes both temporary and permanent files. If the family is the DL JOBS family, then Job File header information will additionally be returned.

SHOWOPEN context scripts may only be used by the EVAL command; there is no support for the DO or WHEN commands. Any EVAL command can be passed a

valid disk family name, with optional familyindex, after the full EVAL command. The familyindex should be preceded by the '#' character

Ex.

DISK #3

Some EVAL examples are shown below:

```
EVAL (SHOWOPEN:TRUE) DO (SHOW(TITLE 50,,SECTORS)) [20] CDIMAGE
EVAL (SHOWOPEN:TEMPORARY) DO SHOW_TEMP DISK #3
```

If no family name is provided then SUPERVISOR assumes the systems file family that it uses (as assigned by USE FAMILY..FOR SYSTEM).

If the DL JOBS family is selected then SHOWOPEN will return information about any Job Files that are active. Typically, in the MCP SHOWOPEN response, these appear as 'Job File for NNNNN' entries; for these entries, SUPERVISOR sets the JOBFIL attribute to TRUE and assigns the job number to the JOBNUMBER attribute.

This can then be used to return additional information about the job file owner:

```
EVAL(SHOWOPEN:JOBFIL) DISP (#J:=JOBNUMBER 5,,#J.MX(NAME)) DEV #1
```

SL Context

The SL context allow the retrieval of information for all MCP SL functions from OPAL scripts using EVAL. The collection of SL attributes currently available can be seen in the response to a HELP ATTR =:SL command.

SL context scripts may only be used with the EVAL or DO commands; in the latter case, the SL function name must be passed as a parameter.

For example, a simple scan of all SL functions whose name begins with the letter 'E':

```
EV (SL:FUNCTION Hdis "E") DO (SHOW(FUNCTION 17,,CODEFILETITLE))
```

To show which SL-ed functions have non-resident codefiles:

```
EV (SL:NOT PD(CODEFILETITLE)) DO (SHOW(FUNCTION 17,,CODEFILETITLE))
ONCRPC SUPPORT      *SYSTEM/ONC/RPC SUPPORT ON DISK
NCSDB SUPPORT      (NAU) SYSTEM/NCSDB/LIBRARY ON DISK
METAWEBLINK        *METALOGIC/WEBLINK ON DEV
MCPSUPPORT          ">> CURRENT MCP <<"
LANPRT SUPPORT      *SYSTEM/LANPRT/SUPPORT ON DISK
JAVACOMSSUPPORT     *SYSTEM/JAVACOMSSUPPORT ON DISK
HLCNTLISUPPORT      *SYSTEM/HLCN/TLISUPPORT ON DISK
```

MCPSUPPORT can be excluded from the EVAL by changing the in-line:

```
EV (SL:NOT MCPLIB And Not PD(CODEFILETITLE)) ...
```

For a standalone DO:

```
DO (SL: (SHOW(FUNCTION 17,,LIBRARYMIXLIST))) TADSSUPPORT  
TADSUPPORT      12345,12444
```

Where LIBRARYMIXLIST returns the mix number(s) of the SL-ed library. This can then be used to determine all SL-users of the codefile. Note that the same codefile instance can be assigned to multiple SL-ed functions or the library may be SHARING=PRIVATE; either condition means that LIBRARYMIXLIST can return multiple mix numbers.

SMTP Context

The SMTP Context may be used to process RFC822 messages from several sources.

An SMTP WHEN processes RFC822 messages received by the SMTP Server from the network using the SMTP Protocol.

An SMTP DO processes RFC822 messages received from the POP Server using the POP3 Protocol, or by scanning a directory on disk for files with a MIMETYPE (FORMID) of "message/rfc822".

SMTP Server - WHEN <smtp situ> DO <smtp odts>

The SMTP Server is implemented by SL SMTP = *METALOGIC/SMTP. It is a DSS and it is initiated by the MCP when a connection is made on Port 25. It may be installed by RUN *METALOGIC/SMTP.

When an SMTP connection is opened, the SMTP Server attempts to link to a SUPERVISOR SMTP Context WHEN. If there are no SMTP WHENs running, the linkage fails, and the incoming mail is rejected.

If there is an SMTP WHEN running, the SMTP Server receives the incoming mail, in the (QUARANTINE) area, but before finally accepting it, sends an SMTP Object to the WHEN. The SMTP Object contains the attributes of the RFC822 message (HELP ATT = : SMTP), which includes the Envelope, the Headers and a reference to the Mail Body.

If the SMTP OPAL code decides to accept the mail, it is responsible for delivering it to the recipients, or forwarding it using MAIL if relaying, and then issuing a RESPOND(SMTP,"OK") command.

The SMTP OPAL is also responsible for delivering a failure notification to any recipients for whom mail was not able to be delivered.

The SMTP OPAL can reject the Mail using a RESPOND(SMTP,"ERR"), or RESPOND(SMTP,"<statusCode> <status message>"). In this case, no delivery failure notification is needed.

POP Client - DO <smtp odts> [<limits>] pop://<popserver>

The POP Client is implemented by SL POPCLIENT = *METALOGIC/POPCLIENT.

When a DO is done, the POP Client uses the pop: URL to open a connection to a

POP Server.

An RFC822 message retrieved from the POP Server is sent to the DO as an SMTP Object, as described above. The <limits> may be used to request that the messages be deleted from the POP Server (DELETE), and to limit the number of lines of each message retrieved (TOP=<lines>).

POP Directory - DO <smtp odts> <flex specification>

The POP Client also provides the capability to search for RFC822 messages on disk. The <flex specification> is provided to FLEX in the same way as a SUPERVISOR PD context. Each RFC822 message selected is returned to the DO as an SMTP Object.

RFC822 Message

An RFC822 message may be thought of as a type of file container. It contains the mail headers ("headers.txt") and the mail body ("body.txt"), which may be structured with additional attached files according to MIME.

MAILIOH

Access to an RFC822 message is provided by the CopyWrite MAIL I/O Handler, SL MAILIOH = *METALOGIC/COPYWRITE/MAILIOH. This library is used by CopyWrite, but may also be used by programs or OPALs using normal File I/O. File "#0." is a file which contains the Directory as for other IOHs.

FLEX

The RFC822 message may be explored using the FILES command of *OBJECT/FLEX, or using a SUPERVISOR PD context. There are always at least two files, one called "headers.txt" and the other "body.txt", and there may be other files which represent the MIME attachments.

For Example:

```
U Flex

Files (*)= IN Mail/20120818/230904/4726904811 on Nucleus

*"headers.txt" : DATA >3<

*"body.txt" : DATA >1<

#Found 2 files [from 2 files]
```

File Attributes

An RFC822 message is contained within a file, and these MCP file attributes are used to store information which may be useful when searching.

PRODUCT	- Contains the From:
NOTE	- Contains the Subject:
RELEASEID	- Contains a UUID which uniquely identifies the message.

This is used by POP Clients to detect duplicate messages.

USERINFO - Set to 0 by SMTP. It is set to a TIME(6) TimeStamp when the file has been retrieved by a POP Client, to indicate that it is no longer a candidate for download by the POP Server.

CopyWrite

Files may be copied out of an RFC822 message using the LIBRARY/MAINTENANCE extensions provided by CopyWrite. For Example:

```
WFL Task T;T(File Mail(  
    Title=Mail/20120818/230904/4726904811 on NUCLEUS));  
COPY *"headers.txt" As H From Mail(CD,HostName=Local,LockedFile)[T]  
    #4850 <1> *"headers.txt" copied as (NFT)H to NUCLEUS
```

OBI

The RFC822 messages can be explored using OBI, providing Browser access to the headers and attached files.

POP Server

The POP Server is implemented by SL POPSERVER = *METALOGIC/POPSERVER. It is a DSS and it is initiated by the MCP when a connection is made on Port 110. It may be installed by RUN *METALOGIC/POPSERVER.

The POPSERVER is independent from SUPERVISOR.

A client logs onto the POP Server using their MCP UserCode/Password. The POP Server scans their MAIL/= directory for any files which have not been downloaded (FILES MAIL/= ON <family> WHERE USERINFO = 0) and builds a list of the files in the mail drop. The POP Client may then retrieve the RFC822 messages or information about them and it may also delete them using the POP3 protocol.

File Naming Conventions

RFC822 messages received from the SMTP Server are stored under the MAIL/= directory.

RFC822 messages retrieved by an OPAL POP Client are stored under the POP/= directory, unless only a sample was retrieved using the TOP=<lines> limit, in which case they are stored under the TOP/= directory.

SMTP Server MAGUS Configuration Options

SMTP_USERCODE

The SMTP_USERCODE is the name of the usercode which is where

incoming mail is quarantined before being rejected or delivered to the recipients.

If not specified the default is QUARANTINE.

The FAMILY statement of the usercode defines where the files will be stored. If the usercode is not defined, the files are stored on the same family as the SL SMTP.

SMTP_AUTH

If SMTP_AUTH is set to "NONE", then no authentication will be required. This means that mail from any source will be received, and passed to the SMTP OPAL context, which can decide what to do with it.

SMTP_GATEKEEPER

If SMTP_GATEKEEPER is set to "ON", then a USERDATA RU check will be performed before a MAIL FROM command is accepted. If SMTP_AUTH = NONE, then the SMTP_USERCODE is used as the Remote UserName, otherwise the authenticated UserName is used.

For Example:

```
+ RU QUARANTINE OF *ANYHOST
```

Security

This first release is intended only for use on a private network since only AUTH LOGIN,PLAIN are fully implemented.

AUTH CRAM-MD5 is implemented but no secure method of keeping the shared secret has been implemented.

The intention is for a later release to provide a secure connection using SSL/TLS.

Operations

-The default Port for the SMTP Server is 25, but this may be changed by modifying the TARGET attribute of METALOGIC/SMTP to a value > 1024.

-The default Port for the POP Server is 110, but this may be changed by modifying the TARGET attribute of METALOGIC/POPSERVER to a value > 1024.

-The POPSERVER, POPCLIENT and SMTP processes will terminate if sent a HI 99 and there are no open connections.

SUPERVISOR DEBUG

TT DEBUG + 38 will set SMTP debugging which is written into the SUPERVISOR Log.

Mail Clients

These Mail Clients have been used in testing,

ThunderBird (12.0) Mail (5.2 OS X 10.7.4) OutLook Express 6 (Windows 2000)

STATIONS Context

Information returned includes detailed information about physical stations (as declared in the NIF) and MCP pseudo stations used to provide sessions for all MCS activity such as CANDE and COMS.

The STATIONS context accepts two subcontexts called NIF and PSEUDO allowing an EVAL to search either datacom station subset. In reality, the PSEUDO subcontext is of more interest since these stations are generally used by COMS and CANDE sessions. By default, if no subcontext is present, EVAL will return all station information.

Examples:

```
EVAL (STATIONS=PSEUDO:TRUE) DO (SHOW(LSN 5, ,STATIONNAME 40, ,MCSNAME))  
EVAL (STATIONS:VALID(CURRENTWINDOW)) DO (SHOW(LSN, ,CURRENTWINDOW))  
DO (STATION:SHOW(LSN, ,LASTLOGONDAY)) 16384
```

Note that the MCP allocates pseudo stations in two different LSN ranges; the first range appears immediately after NIF-declared physical stations and is used by DATACOMSUPPORT for its own station allocation.

The second range is an expanded pool of pseudostations, allocated in blocks of 100 stations and starting at LSN 16384, which is controlled by the MCP.

The EXTERNALSTATUS attribute, if valid, provides additional information provided by Protocol Specific Handlers (PSH), such as CCF and TELNET, and will show connection details, IP address etc.

If the station is active with COMS, additional information such as current COMS window and MCS name plus usercode, accesscode and chargecode if present.

Please see HELP ATTR=:STATIONS for a complete list of available attributes.

VL Context.

This context is only valid on systems where the Cataloging option is set. It allows the volume library of the catalog to be scanned.

It is possible to pass the name of an alternate CATALOG file to an EVAL context script using the same syntax as LOG EVALs:

```
EVAL (VL:TRUE) [TITLE=(TEST)SYSTEM/CAT210711 ON DEV] DO VL
```

In this case, SUPERVISOR will extract the Volume Library from the non-live CATALOG and process all volumes found. Further, if a FLEX STATISTICS file is available with the same directory as the alternate CATALOG then that STATISTICS file will be used. In the above case, the STATISTICS file would have to be named:

VTContext.

The VTContext is intended for use with the SUPERVISOR Remote File Scripting capability, which is implemented by the OPAL WFL function.

When a program writes to a Remote File, the message is interpreted by the TD830 Terminal Emulator and presented on the Screen. When Scripting, the Terminal Emulator function is provided by the VTContext.

A message sent to a Terminal through a Remote File is received within the SUPERVISOR JOB Context as a WFLOUT message. This message can be sent to the VTContext to be interpreted.

There are 2 Interfaces to the VTContext.

DO I EVAL <odts>[EDIT IN SL VTCONTEXT] '<td830 message>'

The EDIT interface interprets a message destined for a TD830 Terminal Emulator. The text and control codes are interpreted to create a Virtual Screen.

The <td830 message> must be enclosed in single quotes (') or double quotes ("), and may be preceded by a <string code>, either 7 for ASCII or 8 for EBCDIC.

When the VTCONTEXT has loaded the <td830 message> it sends Message Objects to the ODTs. The type of Message Object is given by the MSGTYPE attribute. %

These MSGTYPEs may be received.

"LOADED" - The <td830 message> was loaded and the Virtual Screen and its attributes are available.

"STARTFORM" - Only sent if the <td830 message> contains a valid formed screen and the ODTs was started by an EVAL.

"FIELD" - If a STARTFORM was received, it is followed by a FIELD message for each field in the form. The index of the first field is 0.

"ENDFORM" - Indicates the end of FIELD messages.

"TRANSMIT" - This message indicates that an ESC ((Transmit Page) command was embedded in the <td830 message>. The MSGTEXT contains the text that would have been transmitted at that time.

"READSCRATCHPAD" - This message indicates that an ESC RTaaaann (Read ScratchPad Memory) command was embedded in the <td830 message>. The MSGTEXT contains the scratchpad memory read from the configuration.

"WRITESTATUSLINE" - This message indicates that the <td830 message> wrote to the Status Line, the text of which is in MSGTEXT. The STATUSLINE attribute can be read at any time to retrieve the current Statue Line text.

"VERSION" - This message indicates that an ESC sp V (Transmit Version) command was found in the <td830 message>. The MSGTEXT contains the version information, so that it may be transmitted using the OPAL Transmit Function.

"ASTERISKS" - This message indicates that an ESC RC (Transmit Asterisks) command was found. The MSGTEXT contains the uniquely crafted asterisks.

This is an example of a VTContext ODTs using the MSGTYPE.

```
Define + ODTSequence VTF_HELP(CUSTOM):
    % Called for each Field in the Instructions Form
    Case MsgType Of
    Begin
    "STARTFORM":
        $$:=FillIn("0","FILES");
        Display(SaveAs("FLEXFILESHELP","20"));
    "FIELD":
        Display("[",FieldIndex,"] ",LocationOf(FieldStart),"..",
            LocationOf(FieldEnd),"(",FieldWidth,") ",
            FieldType, ",Text=",FieldText);
    "ENDFORM":
        $FormData:=FormData;
    Else;;
    End;
\
```

These attributes are defined for the EDIT Interface of the VTContext.

ATTLIST

Returns a list of the Attributes in the EDIT Interface.

COLUMNOF

`COLUMNOF(Index)` returns the `COLUMN (1..COLUMNS)` of the character at `Index`.

`COLUMNS`

The number of Columns `1..COLUMNS` in the Virtual Screen.

`COPY`

The `COPY(START,END)` copies the data beginning at the index given by `START` up to and including the index specified by `END`.

`CURSOR`

The index `0..PAGESIZE-1` of the Cursor.

`FIELDEND`

The index `0..PAGESIZE-1` of the last data character in the current or specified Field.

`FIELDINDEX`

The Index of the Field in the Form `-1..FIELDS-1`.

`FIELDS`

The number of Fields `0..FIELDS`.

`FIELDSTART`

The index `0..PAGESIZE-1` of the last data character in the current or specified Field.

`FIELDTEXT`

The text in the current or specified Field.

`FIELDTYPE`

The type of the current or specified Field.

`LEFT` - Left Justified Unprotected,

`RIGHT` - Right Justified Unprotected,

`PROTECTED` - Transmittable Protected

`FIELDWIDTH`

The width in characters of the current or specified Field.

`FILLIN`

`FILLIN(FIELDINDEX,TEXT)` fills in the specified field with the `TEXT` supplied.

`FIND`

The FIND(TARGET,START) Method searches the Virtual Screen, starting at the index given by START for the specified TARGET. It returns the Index 0..PAGE_SIZE-1 of the TARGET or -1 if Not Found. It searches for an exact match.

FORM

Returns TRUE if the Virtual Screen is in Forms Mode, otherwise EMPTY.

FORMDATA

Returns the Form data that would be sent to the Host were the Transmit Key pressed with the Cursor at the Home Position.

HELP

Returns Attribute Information for the VTContext.

LOCATIONOF

LOCATION(Index) returns the location of the character at Index as a Co-ordinate (ROW,COLUMN).

MSGTEXT

The Object Message types TRANSMIT,READSCRATCHPAD,WRITESTATUSLINE, VERSION and ASTERISKS have associated text, which is returned as EBCDIC.

MSGTYPE

The Type of Object Message is one of LOADED,STARTFORM,FIELD,ENDFORM,TRANSMIT,READSCRATCHPAD, WRITESTATUSLINE,VERSION,ASTERISKS.

PAGES

The number of Pages configured.

PAGE_SIZE

The page size (ROWS x COLUMNS) of the Virtual Screen.

ROWOF

ROWOF(Index) returns the ROW (1..ROWS) of the character at Index.

ROWS

The number of rows 1..ROWS in the Virtual Screen.

SAVEAS

SAVEAS(TITLE). The current page is formatted as HTML and saved in a file with the specified TITLE. If any error occurs, the function returns ERROR:<reason> otherwise EMPTY.

SPCFY

Location. SPCFY(INDEX) moves the Cursor to Index, returns the escape sequence for that location. The escape sequence is returned in EBCDIC.

STATUSLINE

STATUSLINE returns the text of the Status Line.

The 2nd VTContext Interface is BUILD.

DO <odts>[BUILD IN SL VTCONTEXT]

The BUILD interface may be used to create a Virtual Screen, using Text and TD830 Control Codes.

When the VTCONTEXT has loaded the <td830 message> it sends a LOADED object (MSGTYPE="LOADED").

These attributes are defined,

ATTLIST

Returns a list of the Attributes in the BUILD Interface.

BLINK

BRIGHT

CLEAREOL

CLEAREOP

Return Control Codes.

COLUMNS

The number of Columns 1..COLUMNS in the Virtual Screen.

CR

DELETE

DELETELINE

DOWN

Return Control Codes.

EDIT

Take the TD830 formatted message in P1, translate it to ASCII and send it to the EDIT Interface. Return the Virtual Screen created in EBCDIC.

ENDFIELD or EFIELD

ENDHIGHLIGHT

ENTERFORM

ESC

EXITFORM

FF

Return Control Codes.

GOTO

GOTO(ROW,COLUMN) returns the Control Codes for moving the Cursor to the specified location.

HELP

Returns Attribute Information for the VTContext.

HOME

HOMECLEAR

INSERT

INSERTLINE

LEFT

LF

LEFTFIELD or LFIELD

MOVELINEDOWN

MOVELINEUP

Return Control Codes.

MSGTYPE

The Type of Object Message is LOADED.

PROTECTEDFIELD or PFIELD

REVERSE

RIGHTFIELD or RFIELD

RIGHT

ROLLDOWN

ROLLUP

SECURE

UNDERLINE

UP

Return Control Codes.

WRITE

Returns the text provided in P1.

There is an example of using the EDIT Interface to script *OBJECT/FLEX in the file *SUPERVISOR/VTCONTEXT/EXAMPLE, which is available from Metalogic on request.

WHEN Context

This is primarily an event-based context that allows SUPERVISOR to monitor all WHEN and DO activities such as sessions starting and terminating, with reporting for those that have faulted. The full set of attributes can be viewed using **HELP ATTR=:WHEN**.

WHEN context OPAL programs may be used with the WHEN command (the type of event is designated by the EVENTTYPE attribute) and also with EVAL. For HTTP WHEN programs, a special EVENTTYPE with the mnemonic WHTTPBUSY has been added; this allows SUPERVISOR to detect a browser HTTP client request that has become queued because all the active HTTP WHENs are currently busy.

Both the COUNT and OBJECTS functions in OPAL will accept WHEN context calls, returning a list of slot numbers in the latter case. WHEN programs may also be executed standalone, using the DO command, by passing a valid slot number as parameter.

```
WHEN (WHEN:TRUE) DO WHENODTS
DO WHENODTS 1,22,23
EVAL (WHEN:ODTSNAME INCL "META") DO (SHOW(SLOT,,SITUNAME,,ODTSNAME))
```

The WHEN context may take a TIME modifier as part of a subcontext assignment (as with MX and PER) allowing such OPAL programs to poll the current WHEN list using an user-specified DELAY. However, unlike MX and PER, a time-based WHEN slot will still also be triggered by normal WHEN events created when a slot is activated, faulted or terminated. For a timed poll, the information for each slot passed to the WHEN has a special EVENTTYPE category called WTIMEOUT.

Example:

```
DEFINE + SITUATION W(TIME,WHEN) :
SITUCONTEXT="HTTP"
```

Two new WHEN context attributes have been added. SITUATION gives the name of the situation with a prefix of the command used to invoke it. Ex EVAL MY_SITU. ACTIVITY give the name of the ODTs,COMMAND or DISPLAY with a prefix of the command used to invoke it. Ex DO MY_ODTS.

ODTSequences of type DEFINE now require a type before the name and wildcards are no longer allowed. EX DO MYDEF ODTs MY_ODT. In order to be able to accept input derived from the new attributes above DO MYDEF DO MY_ODT is also allowed. DO is a synonym for ODTs WHEN or EVAL are synonyms for situ. If the parameter passed refers to an Inline Opal then a skeleton entry is returned with only the TYPE,Name and Source attributes valid.

Execution of ODTSequences

An ODTSequence may be executed with a DO, EV or WHEN command, except for COMPLETED ODTSequences which must be tied to SITUations of the same type by a WHEN command.

ODTSequences of the SYSTEM context do not need to have their objects supplied. All other ODTSequences have to be supplied with the objects that they are to act upon. For example a MX context ODTs needs a list of the mix numbers that are to be acted upon. PER ODTSequences require specification of the list of unit numbers and unit types. These are supplied by their SITUations if executed by a WHEN or EV command. In an explicit DO command, the lists must be supplied in the parameter. For each entry in the parameter list, all statements, including any WAIT statements, are executed sequentially. After all statements have been executed for an item of the list, SUPERVISOR restarts the execution of the ODTSequence from the first statement for the next item in the list. When all items of the list are exhausted, the ODTSequence completes.

The parameter for a MSG context ODTSequence consists of the text that the message would contain and is available in the TEXT attribute. Because the text may contain any character, there is no syntax for giving a list. However, this context is very useful because it effectively gives the ability to have an ODTs with a STRING parameter. For example, this is a simple ODTSequence that will send a user-specified message to SYSTEM/COMS

Example

```
TT DO TO_COMS READY ET25
```

```
TT DEFINE + ODTSEQUENCE TO_COMS (MESSAGE) :  
    ODT (COMSMIXNO, " SM ",TEXT)
```

While SUPERVISOR is executing an ODTSequence, it evaluates all <ODT string> statements, SHOW, and DISPLAY statements, and holds the results in local storage until either the last statement has been executed or a WAIT or WHEN statement is encountered. At this point all held messages and system commands are displayed or passed to the system. This is done to avoid changing the "state" of what is being observed.

For example, if an MX ODTSequence first DS-ed a task and then executed a DISPlay, there would be no program on which to display the message. Worse, execution of ODT commands might change the value of some ATtributes used in

constructing <ODT string>s.

OPAL Compiler Options

The OPAL compiler, like the standard Unisys compilers, can produce a line printer listing of the source (SET LIST) or the byte code (SET CODE). The VERSION option puts a number into the SCHEDULE with a definition. It can be seen in the DEFINE ? command response.

CODE is not valid for MEMOs.

Program Sources

The source of the definitions can be kept within SUPERVISOR or in normal source files of FILEKIND JOBSYMBOL. If the text is to be left in SUPERVISOR's OPAL directory for later modification a colon ":" is used, followed by either the text of a MEMO or an OPAL program valid for the type of program and the context. The OPAL program is specified in a free format but keywords and strings cannot be split across the end of a line. More than a screen of input can follow the colon if the DEFINE command is in an ENTER file.

SUPERVISOR preserves indentation as long as there is no text on the line following the colon.

```
TT DEF + ODTs TEST:
    IF WM = "XYZ" THEN
    BEGIN
        ..
        ..
    END
```

will have indentation preserved.

If, however, the same OPAL program were to be entered as follows

```
TT DEF + ODTs TEST: IF WM = "XYZ" THEN
    BEGIN
        ..
        ..
    END
```

the text following the colon on the first line will be moved to the next line down and a subsequent DEFINE ? to list the program source would now show:

```
TT DEF + ODTs TEST:
    IF WM = "XYZ" THEN
    BEGIN
        ..
        ..
    END
```

SUPERVISOR automatically indents these additional lines by 2 characters.

It is strongly recommended that program text or comments start on the next line after the DEFINE header, to aid clarity and avoid this behaviour.

DEFining FROM files

Programs can also be DEFINEd from a file. The <file title> in the "FROM <file title>" syntax must have an ON <family name> part where appropriate.

For example:

```
TT DEFINE + SITUATION LONG_FILE FROM (OPS)OPAL/LONGFILE ON DEV
```

For these programs, SUPERVISOR keeps only the name of the file and the code internally and there is no practical limit to the size. The file should contain only the OPAL program itself, with no TT commands. The format can be free form but, as in direct input, strings and keywords must not be split across record boundaries.

With the "AT <host name>" syntax, a file can be loaded from another host on a BNA network that allows OPAL program storage to be centralized and kept consistent across many machines. Any attempt to read a foreign file will fail if the correct ALIASES are not pre-DEFINEd on the two machines. SUPERVISOR has no special privileges across the network.

Example:

```
TT DEF + SITU TEST FROM (OPS)OPALS/TEST ON DEV AT STIRLING
```

Only one definition may appear per file and must contain only the body of an <OPAL program> or MEMO text. If the number of lines is less than or equal to 24, the text of the MEMO or OPAL program is stored in the SCHEDULE as if it originated from a system console, otherwise the text from the file is not placed into the Dictionary, only the text "FROM <file title>". The modification of such DEFINES can only be made by altering the source and re-entering the DEFINE+ command.

The use of DEFINE..FROM is not the preferred way of maintaining and loading long OPAL programs in the SCHEDULE. The ENTER command can be used to compile and save multiple OPAL programs, storing sources and code directly into the SCHEDULE. Long OPAL program sources held in the SCHEDULE can be viewed, but not amended by a DEFINE ? command.

Using the FOR modifier with DEF

If a FOR clause precedes the DEF command, or if the terminal is running with an active TERM USER specification, the identifier is entered into SUPERVISOR with a usercode attached. This OPAL program can then only be run, listed or deleted by that usercode (or by a privileged usercode if the creating usercode is not privileged).

SUPERVISOR will give back a "FOR <usercode>" prefix in response to a "DEFINE ?" command if the definition is owned by a usercode, unless the terminal has that usercode as its TERM USER. TERM USER is a standard system command on the system consoles, and SUPERVISOR permits TT TERM USER on

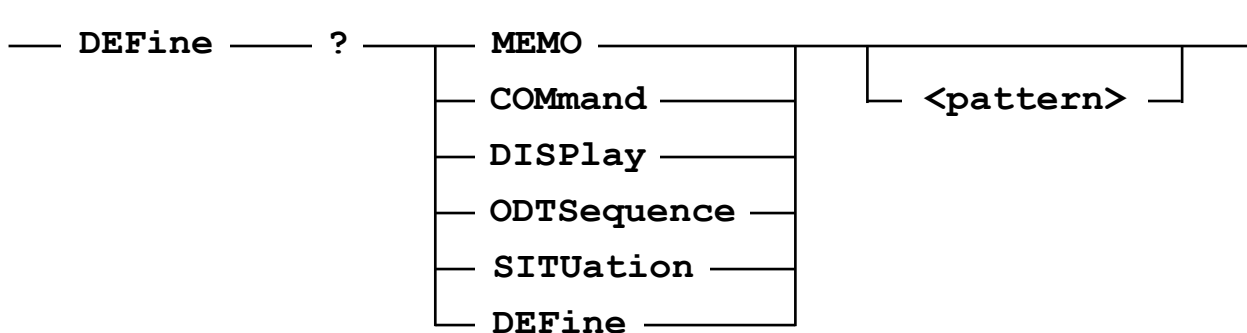
its REMOTESPOs and WINDOW stations. Definitions made on a terminal with an active usercode are entered into the OPAL dictionary under the terminal's usercode.

Examples

```
TT DEF + ODTs TAPELIB (TAPELABEL) : TAPERECORD ;
TT FOR USR/PW DEF SITU COND (PER=MT) FROM (USR) COND ON PKB
TT DEF+DISP RN (TAPELABEL) : (IF REEL<3 THEN "NEXT" ELSE "LAST")
```

Note that the **DEFINE** command is also understood directly by OPAL in the stand-alone OPAL compiler, where OPAL programs can be developed and compiled in CANDE (just like ALGOL or WFL) for syntax prior to testing within the SUPERVISOR environment. Once a workfile has been clean compiled, it can be ENTERed through SUPERVISOR for testing. See the OPAL manual for more details.

DEF ? : Interrogating the OPAL Directory



SUPERVISOR keeps a directory of OPAL programs available. The directory is like a simplified file directory with only 2 levels – the usercode and one file identifier. The identifier in OPAL is similar to that in WFL and allows a numeric first character and embedded underscore characters, e.g. 1_2 is a valid identifier.

The DEF ? syntax is used to list these internal directories. There are 4 kinds of OPAL program: SITUations, ODTSequences, COMmands and DISPlays. Each directory is kept separately. A partial directory list can be obtained by using wildcards in <pattern> in the railroad diagram above. For example, using a trailing "=" means that the response should only include those OPAL programs which match the characters up to the "=".

Example

```
TT DEF ? SITU
```

means list all visible SITUations

```
TT DEF ? ODTs XY=
```

means list all ODTSequences that begin with the letters "XY".

The key word DEF, in place of MEMO, SITU, ODTs, or DISP means that the directory entries should be reported independently of the kind of definition.

```
TT DEF ? DEF XY=
```

means list all definitions that begin with the letters "XY".

```
TT DEF ? DEF ZZ
```

means list all OPAL programs called "ZZ".

Further filtering may be achieved using an optional modifier to limit the search to a particular context, similar to the HELP ATTR equivalent.

For example:

```
TT DEF ? DEF =:PER
```

```
TT DEF ? ODTs A=:TAPEDB
```

```
TT DEF ? SITU =SS=:FILECLOSE
```

In the above cases, only those DEFINES that belong to the relevant context will be returned in the response list.

The responses are sorted alphabetically, and return the name and type.

```
DEF ? DEF BOB_A=
-- SITUATIONS FOR * ----- Opal --- Last Used ----- Created ---- Ver
BOB_A          MX=A          530.12  08:37,18/04/08*08:35,18/04/08 3
BOB_ABEND      C            520.15      Never      13:12,01/06/06
BOB_AC_EX10    JOBREJECT    520.15      Never      13:12,01/06/06
BOB_AC_EX8     U            520.15      Never      13:12,01/06/06
BOB_AC_EX9     LOG=1,7      520.15      Never      13:12,01/06/06
BOB_AUDIT      FILECLOSE    520.15      Never      13:12,01/06/06
BOB_AX         MX=W          520.15      Never      13:12,01/06/06

-- ODTSEQUENCES FOR * ----- Opal --- Last Used ----- Created ---- Ver
U BOB_A        MIX            530.12  c08:37,18/04/08*08:35,18/04/08 5
BOB_ABEND      COMPLETED    520.15      Never      13:12,01/06/06
BOB_ABORT      TAPEDB        520.15      Never      13:12,01/06/06
BOB_AC_EX1     SYSTEM        520.15  08:38,18/04/08 13:12,01/06/06
BOB_AC_EX10    JOBREJECT    520.15      Never      13:12,01/06/06
BOB_AC_EX2     SYSTEM        520.15      Never      13:12,01/06/06
BOB_AC_EX3A    SYSTEM        520.15      Never      13:12,01/06/06
BOB_AC_EX3B    SYSTEM        520.15      Never      13:12,01/06/06
BOB_AC_EX4     SYSTEM        520.15      Never      13:12,01/06/06
BOB_AC_EX5A    SYSTEM        520.15      Never      13:12,01/06/06
BOB_AC_EX5B    SYSTEM        520.15      Never      13:12,01/06/06
BOB_AC_EX6A    MESSAGE       520.15      Never      13:12,01/06/06
```

The first column of the response is reserved for a marker applicable only to ODTSequences to warn the User that the Opal code may be unsafe, this is followed by two columns giving the names matching the search criteria and the context of each.

In the example screen shown above the ODTSequence BOB_A is marked as unsafe Opal code and could not be invoked if the **STRICTODTS** option was SET. | Unsafe constructs are discussed further in the **METALOGIC OPAL Programming Reference Manual**.

The **Opal** column reports the version of the Opal compiler used to compile the Opal. For Opals compiled before version 400.29 of Supervisor, this value has a suffix of 'S' to indicate that this was the version of Supervisor running when the Opal was compiled.

The **Last Used** column is the date and time when the Opal was last run. (Time started if in use at present, time stopped if not running). An asterisk follows this column if the Opal is currently in use. A letter 'c' preceding this column for an ODTSequence indicates that the ODTs was called by using the 'CALL DO ' statement from another ODTs, or if it is active then the time it was called by a SITUation.

The **Created** column shows the time and date when the Opal was compiled.

The final column, marked 'Ver' reports a user DEFINEd version which can be set using the VERSION <integer> option of the DEFINE + command.

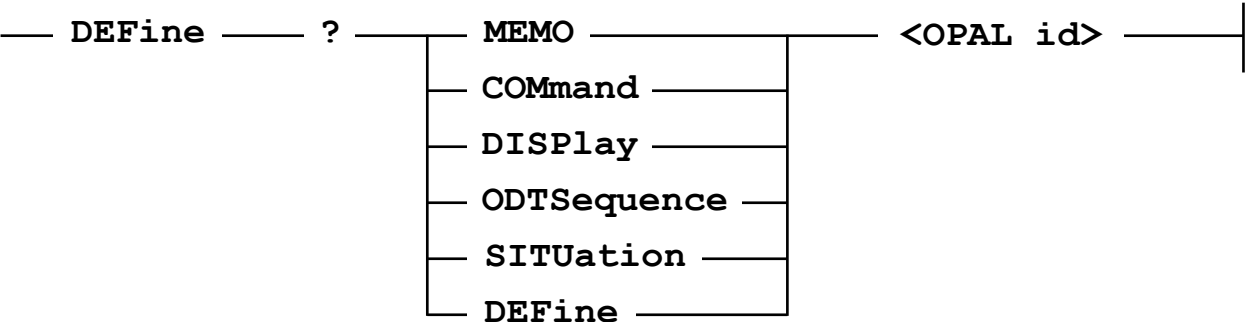
Example:

```
TT DEF + ODTS BOB_DIE VERSION 6:  ....
```

To get information on OPAL program names under a usercode, either use a FOR clause in front of the DEF command or use a TERM USER command before entering the DEF command.

```
TT FOR USER/PW DEFINE ? ODTS TEST_ =
```

DEF ? : Listing a Program



A program can be listed on a terminal by using the DEF ? variant of the command. The DEF + format is used in the response to make it easier to modify the program on a screen.

When using the DEF ? command for a multi-page OPAL, the usual DEFINE + ODTSEQUENCE, displayed in the home position of the screen, will be over written by TT + followed by a series of percent (%) characters. This is to inform the user that the program is a "long" OPAL and avoids accidental update. Such OPALs must be amended using TT SAVE, amending manually using CANDE, or Programmers Workbench and then passed back to Supervisor using [ENTER](#).

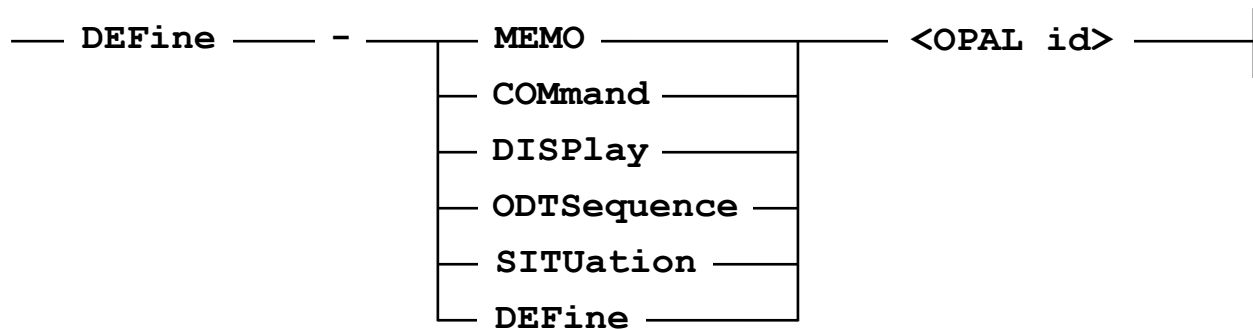
If a FOR clause was used when creating the program, it must either be used to list it, or the listing should be done to a terminal with the correct TERM USER set.

If the program was originally created using a DEF + .. FROM command, the program listing is not kept within the SCHEDULE and this variant will only return the text of the original DEF + .. FROM command. i.e. the name of the file where the source is stored.

The use of DEF + ..FROM is not recommended, since it is easy to lose the original source and be left with a routine which cannot be modified.

If the program is a MEMO, it may be viewed using the MEMO command. If it is an OPAL program, it may be viewed by changing the program type to MEMO, re-inputting the DEF +, then using the MEMO command as before.

DEF - : Deleting a Program from the Directory



A program is deleted using the DEF – syntax. Note that if the program was created using DEFFROM a file then the file is not removed, only the identifier and its associated compiled code are removed from the directory of OPAL programs in SUPERVISOR.

It is not possible to delete a 'directory' of OPALS by using wild cards; the DEFINE – command only operates on a single DEFINE name at any time.

If the selected OPAL program is 'locked' under a usercode, a 'FOR' modifier must prefix the DEFINE -, or a switch of usercode, using the 'TERM USER' command, must be used prior to the delete.

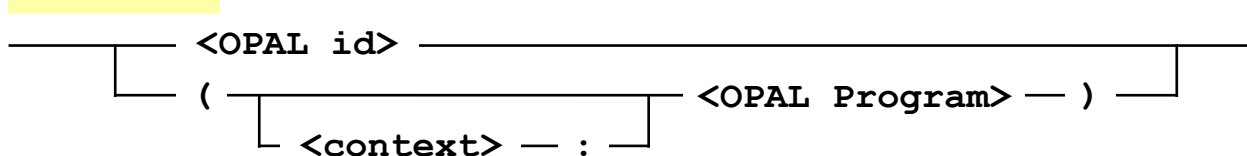
Examples

```
DEF - MEMO OPL_TESTMEM
DEF - DISP OPL_TESTMEM
FOR USER/PW DEF - ODTS META_OPAL
```

IN-LINE OPALS

All SUPERVISOR commands for executing OPAL programs have the option of taking the OPAL from the SCHEDULE, by giving the OPAL identity of an existing program, or placing the OPAL code directly in-line in the command.

<OPAL def>



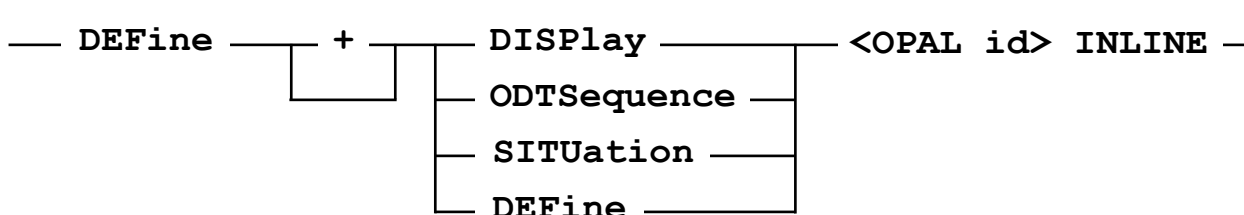
In-line OPALs are considered temporary and are overwritten when another in-line OPAL is entered, OPAL source and code are not written into the SCHEDULE. The DEFINE command gives the ability to view and save the inline OPAL.

The <OPAL def> modifier can appear after any SUPERVISOR OPAL-related command; the OPAL expressions will only be considered as an in-line OPAL if bounded by a pair of parentheses. Commands such as WHEN and EVAL permit mixed in-line and defined OPAL programs if desired.


```
TT / (MCP,/,SI)
TT EVAL (LOGEOJ:NAME INCL "TEST") DO PRINTLOGENTRY
TT WHEN (MX=W:NO "FILE" ISIN RSVP DO (ODT(MIXNO, "DS0"))
TT WHEN GETWAITING DO (RECORD[ALERT]($MESSAGE))
```

An in-line OPAL can be a maximum of 45000 characters. Note that the context is NOT required for a DISPLAY or ODTSEQUENCE in-line if it has been associated with a SITUATION.

Saving



If a Program Type is used, only that type will be saved from the command. If DEF + DEF is used, then an attempt is made to save the Situ and associated ODTs or DISP from the most recent INLINE command, under the given identifier. If the most recent Inline command did not have both a SITU and an Action then an error is returned.

```
EV (MX=A:MCS) DISP (MIXNO,,NAME)
```

DEF ? INLINE

DEF + SITU MCSES INLINE

```
DEF + SITU MCSES (MX=A) :
MCS
```

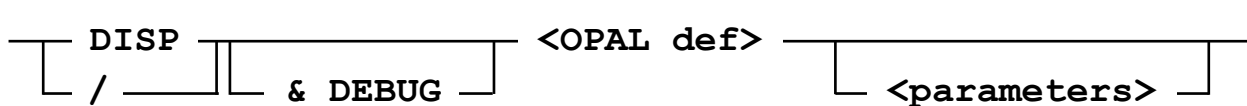
while

DEF + DEF MCSES INLINE

is then equivalent to

```
DEF + SITU MCSES (MX=A) :
MCS
plus
DEF + DISP MCSES (MX) :
MIXNO, , NAME
```

DISPlay Invocation Command



A <DISPlay invocation> invokes an OPAL DISPlay, and sends the resulting output to the terminal. The '& DEBUG' option ensures that a debug trace of the executing OPAL will be provided.

Parameters

To execute a stand-alone DISPlay program using this command, only specific OPAL contexts are allowed. Contexts or object classes, such as MX or PER, require a cardinal attribute to be provided i.e. an attribute which uniquely identifies a particular object.

MX parameters

In type MX, this cardinal attribute is the MIXNO and <parameters> must be a <mix number list>

TT / SHOWMIX 9134,1298,1301

PER parameters

In type PER, a <unit number list> can be passed. A <unit number list> consists of an optional unit mnemonic followed by a list of units of that type, in a similar format to system commands. Since all unit numbers are unique SUPERVISOR does not insist on a unit mnemonic.

TT / (PER:RFERRORS) MT 98-103,LP 17,25-27
TT / (PER:RFERRORS) 500,501

MSG parameters

If the object class of the DISPlay is MSG, then any text that appears between <DISPlay identifier> and <ETX> is passed as the "text" parameter to the DISPlay, which is accessible from the MSGPARAM attribute.

```
TT / MSG_TEST THIS IS A TEST
```

TAPEDB parameters

If the object class is TAPEDB, the cardinal attribute is SERIALNO, and <parameters> must be a <serialnumber list>.

```
TT / (TAPEDB:LOCATION, ,NOTE) META03,10567
```

PRINTS parameters

For the PRINTS context, the cardinal attribute is REQUESTNO.

```
TT / SHOWPRINT 16619,16617,16615
```

VL Parameters

If the object class is VL the cardinal attribute is SERIALNO and <parameters> must be a <Serialnumber list>. Information from the MCP's Volume Library and Flex volume statistics will be automatically merged for individual volumes that appear in both environments.

For example:

```
TT DO (VL:SHOW(SERIALNO, ,VLNAME 17, ,BACKUPREFS)) DF0002,CI0005
```

Returns:

```
DF0002 CDIMAGE_A      15
CI0005 CI0005M2007176A 27
```

Note that only a serial number need be used and no volume kind is necessary. SUPERVISOR will automatically interrogate the Volume Library for each serialnumber using the available kinds of DISK, PACK and TAPE and will execute the ODTSEQUENCE if any eligible entry is found. This means that it is possible for multiple calls to be executed for a serial number that has two different volume entries.

USER parameters

USER context routines will accept an usercode list.

For example, assuming the Opal script is running with a privileged usercode:

```
TT / (USER: USER, , FAMILY, ,ACCESSCODELIST) META,FLEX,TEST
```

Returns:

```
META DISK=DEV OTHERWISE DISK IPP/?  
FLEX DISK = DEV OTHERWISE DISK RULES/?  
TEST DISK = VMSA OTHERWISE DISK
```

Invalid usercodes in the parameter list will each generate an error message but not force the DO or DISPLAY to abort.

If a DISPlay is invoked with a parameter list, the generated output for each parameter in the list is queued until the parameter list is exhausted and then returned to the caller in one or more paged outputs.

AFTER parameters

AFTER context programs require a more complex parameter. To retrieve information for an unique record in the schedule, the DO command must be given the schedule day or specific julian ON DATE, the minute of day and the activity number. The Opal KEY attribute provides this access but its use is not intended for general use.

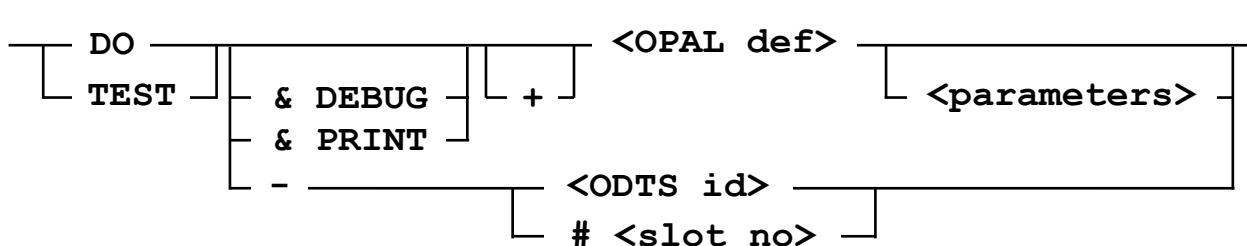
SYSTEM parameters

No parameter is allowed for SYSTEM DISPlays.

Example

```
TT / CANDECOUNTS
```

DO Command



The DO command is used to run or stop an ODTSequence without a controlling SITUation. TEST is used to debug an ODTSequence by turning any statement that would cause an action into a display message. The & DEBUG option invokes a debug trace for this invocation. "DO +" or "TEST +" means that if the ODTSequence is already running the DO is not invoked. Without the "+", it would run in any case.

The & PRINT option forces all SHOW statements to be redirected to printer backup instead of the source station or ODT. The change applies to both the DO and WHEN..DO commands and is preserved after SUPERVISOR restarts and to decompiled SCHEDULE files.

Ex.

```
DO & PRINT (SHOW("This is a test"))  
WHEN SITU DO&PRINT ODTS
```

The DO may be "locked" under a usercode using the FOR modifier.

<parameters> and order of execution are exactly the same as in the [Display](#) Invocation command. Please refer to that command for more details.

Some examples:

```
TT DO (PER:IF INUSE THEN ODT("CLMT",UNITNO)) MT11-15
TT TEST DSNOFIL 1235,1298-1301
TT DO & DEBUG MYOPAL
```

DO Breakpoints

Using the [@..] extension to the DO command, it is possible to force a printed OPAL dump prior to the execution of an individual operator. This feature is ONLY currently permitted with the DO command and may only be executed once. Program execution will continue after the dump has been printed.

The operator can be selected from a DEFINE listing generated using SET LIST CODE after the DEFINE name.

```
DEFINE + ODTs TEST SET LIST CODE:
    $A:"THIS IS A TEST";
    SHOW($A);
```

A fragment of the code listing of the above would be (operator syllable number is the left-hand column):

```
0043 EXIT
0044 SHOW
0045 STOP
```

To force a dump at code syllable 44 (that executes the SHOW statement):

```
TT DO [ @ 44 ] TEST
```

It is very likely that this facility will only ever be used by Metalogic personnel for development and fault debugging purposes.

Please see the [SLOT](#) command for a description of the printed dump listing generated by the breakpoint.

Terminating a DO

"DO - " is used to terminate an ODTSequence that was run with a DO or TEST. Note that the name of the ODTSequence is required and all active DO slots with the given name will be terminated. If the DO is locked under a usercode, a FOR modifier must be used to allow the slot to be unlocked.

An alternate method to terminate a specific DO is to use DO – with the '#<slot no>' modifier where <slot no> is Supervisor's assigned WHEN slot identifier for each WHEN or DO. This mechanism allows *any* WHEN or DO to be easily terminated

without providing the original command.

Examples

```
TT DO - LODU
TT DO - #11
TT FOR TEST/PW DO - #3
```

DO practicalities

Execution of the ODTSequence is subject to a WHEN slot being available. If the number of active WHENs is too great, execution of the ODTSequence will be deferred. A WHEN ? response will show the status as QUEUED.

Furthermore, when the code for the ODTSequence is loaded from the SUPERVISOR Schedule it's associated 'Unsafe' code marker is examined. If the code is marked as 'Unsafe' and the STRICTODTS option (SO+STRICTODTS) is SET, the DO will not be started and an error message indicating that an attempt has been made to execute 'Unsafe' code will be displayed.

Whenever an ODTSequence has within its code an ODT statement, which issues a SUPERVISOR DO command, SUPERVISOR will attempt to DO the ODTSequence under the same session as the invoking ODTSequence. It will also mark the origin to be the same as the origin of the invoking ODTSequence. This means that SUPERVISOR knows where to send any information it may generate, and any responses will appear on the originating terminal.

However, if the origin cannot be identified, the request will be ignored. This can happen where an ODTSequence invokes an EVALUATE with DISPLAY and then stops. In this case the order in which things occur internally precludes SUPERVISOR from identifying the origin of a command (the invoking ODTSequence has "gone away" before the EV is processed by SUPERVISOR).

A way to avoid this is to put a WAIT for a small time (1–10 seconds) after the EV command.

For example:

```
TT DEFINE + ODTSEQUENCE XYZ:
    ODT("TT EV CPUHOG DISP Y");
    WAIT(5)
```

An alternative is to use the TT function:

```
TT DEFINE + ODTSEQUENCE BOB:
    SHOW(TT("EV MCSES DISP MCSES"))
```

The LOG command is a useful aid when debugging ODTSequences as all SUPERVISOR generated commands are written to the LOG file.

Another means of debugging ODTSequences is to use the TEST command rather than the DO command. In TEST mode, no system commands are passed to the MCP; rather they are displayed in a message like:

SUPERVISOR/DO+XXX ODT:<value of ODT string>

The MONITORING option (SO+MONITORING) can also be used to audit or debug ODTSequences.

Example of TEST & DEBUG

**TT TEST & DEBUG OPL_TEST
THE DO WILL BE DONE**

The backup file can be listed with the CANDE BACK command:

```
% #1: *BD/0004257/000TASKFILE ON PACK (Lines: 1-15 of 15)
000001 15:50:51.6  ENTER SETMEUP FROM 20D:00F4:1
000002 15:50:53.3  SLOT 10 ASSIGNED TO DO OPL_TEST SESSION 4580
000003 15:50:53.3  EXIT  SETMEUP
000004 15:50:53.4  ENTER DOODTSEQUENCE FROM 20D:0169:1
000005 15:50:53.5  CODEX=0000,PREG=SMDE,TOS @009 =
800000000000,S=0,STR=(0),STR2B=(0)
000006 15:50:53.6  ENTER OPAL_MACHINE FROM 1DF:02CB:0
000007 15:50:53.6  CODEX=0003,PREG=LTDS,TOS @009 =
800000000000,S=0,STR=(0),STR2B=(0)
```

OPAL CALL DO statement

As described earlier, ODTSequences are usually executed with the DO command, either stand-alone or associated with a WHEN or EVAL command. However, small, commonly used ODTSequence code snippets can be called by other ODTSequences using the OPAL CALL DO statement.

For example:

CALL DO <string expression>

Since an individual ODTSequence can be CALL-ed many times from one or more other ODTSequences, SUPERVISOR optimises CALL DO's by using an internal caching mechanism to hold the code in memory.

The CALL statement suspends the execution of an ODTSequence and immediately passes control to the ODTSequence or DISPlay whose name is the value of the <string expression>. The <string expression> need not be constant and will be evaluated when the CALL statement is executed. The DO variant is used to invoke an ODTSequence while the DISPlay variant will cause an Opal Display program to be executed.

Since the 'Called' code is running in the same slot as the original ODTsequence, all variables of the original ODTs are available to the 'Called' code and vice versa.

If the CALLing ODTSequence is running under a locking usercode, SUPERVISOR will look for an ODTSequence or DISPlay defined under that locking usercode. If none is found, it searches for an ODTSequence or DISPlay defined without a usercode. If no candidate is found, the CALLing ODTSequence will be terminated and an error message displayed.

families of OPAL programs (according to type or name) to be loaded from one ENTER file.

So for example

```
TT ENTER DEF OPAL= FROM SUPERVISOR/OPALS
```

would load all DEFINES in the file SUPERVISOR/OPALS whose id started with the root identifier "OPAL".

The <pattern list> item in the railroad diagram allows multiple selections of program identifiers to be loaded from the same OPAL symbolic file. Each program specified in such lists may be wild-carded.

For example:

```
TT ENTER DEF OPAL=,HL=,STN_ = FROM (META)OPALS/EXAMPLES
```

In the above example, all OPAL programs (regardless of type) which begin with OPAL, HL or STN will be ENTERed from the specified file.

If no wildcard symbols are used in <pattern list> then a direct name match is necessary for the ENTERed OPAL to be compiled.

For example,

```
TT ENTER ODTs WORK_TEST FROM SUPERVISOR/OPALS
```

would load the ODTSequence whose name was **WORK_TEST**.

Currently, both SAVE and ENTER will allow up to 24 identifiers to be used in a program list.

Attempts to exceed this limit will return the error message:

```
TOO MANY ID'S SPECIFIED IN SAVE/ENTER LIST
```

When ENTER REF is used instead of ENTER FROM, there is a difference in the way OPAL programs are stored. If FROM is used, the source is kept within the SCHEDULE; if REF is used instead, only the name of the file that it came from is kept.

An OPAL listed by a DEF ?, after being loaded by an ENTER REF such as:

```
ENTER REF OPALS/BOB ON SYS AT STIRLING
```

will after a DEF ? SITU BOB, return a line of the form:

```
ENTER SITU BOB REFERENCE OPALS/BOB ON SYS AT STIRLING
```

It is possible to hold, directly in the SUPERVISOR SCHEDULE, the source for an OPAL program that is longer than 22 lines. However, the only way such programs may be maintained is by the ENTER and SAVE commands. The maximum size of an Opal program is determined by the size of the source added to the size of the compiled code. This must be less than 2184 30 words segments(Approx 393kB)

When using the DEF ? command for a multi-page OPAL, the usual DEFINE +

ODTSEQUENCE, displayed in the home position of the screen, will be over written by TT NS followed by a series of percent (%) characters. This is to inform the user that the program is a "long" OPAL and avoids accidental update. Such OPALs must be amended using TT SAVE, amending manually over CANDE, followed by ENTER.

Example of making the input file in CANDE:

```
M ENTERFILE J
100TT DEF + SITU BADMT(PER=MT) :
200          RFIOS < RFERRORS*20 \
300TT DEF + ODTs BADMT(PER) :ODT("RW MT",@) ;
400          DISPLAY("BAD MT") \
500TT WHEN BADMT DO BADMT\
SA
```

This is entered with:

```
TT ENTER FROM (MYUSER)ENTERFILE ON MYPK
```

If a file contains several OPAL programs, SUPERVISOR gives a paged response to the ENTER, detailing which programs failed and why, plus a summary line indicating the totals of success and failure.

For example

```
TT ENTER FROM SUPERVISOR/OPALS ON PACK
```

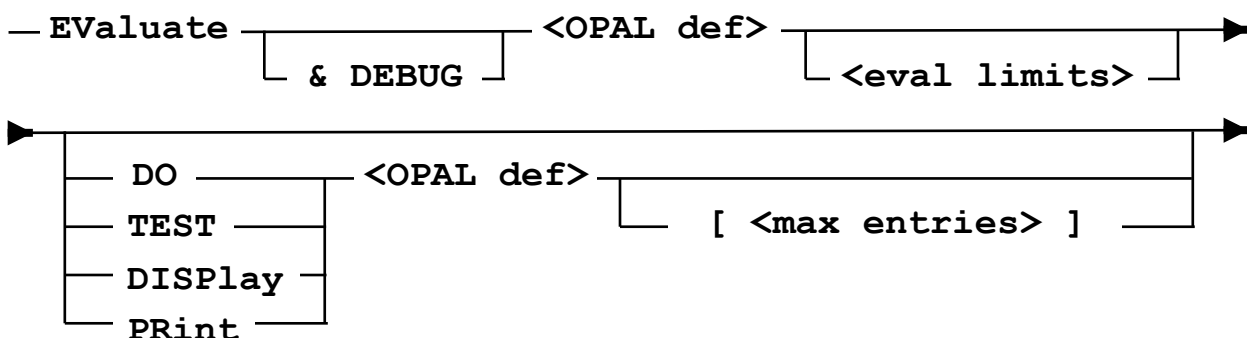
might generate the response

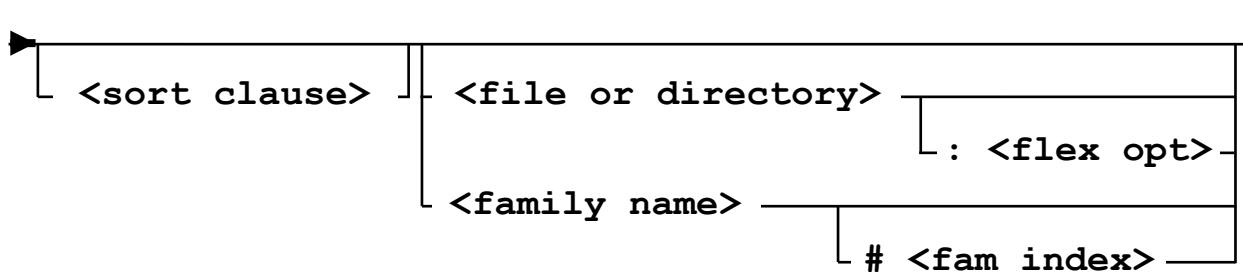
```
ERRORS FOUND IN DISPLAY: DISP_EXAMPLES
ERRORS FOUND IN ODTSEQUENCE:WAITING_ODTS
2 OPALS COMPILED OK, 2 OPALS DID NOT COMPILE
ENTER COMPLETED
```

EVALUATE Command

The EValuate command is used to evaluate, monitor, or terminate the execution of SITUations.

Evaluating a SITUation





This variant of the `Evaluate` command is to execute `SITUations`, which return their results to the terminal, and can also pass them on to an action such as an `ODTS` or `DISPlay`. In object terms, the `Evaluate` invokes the `SITUation` once, which returns `FALSE`, or `TRUE` and a set of objects, which may then be passed, one by one, to an attached `DISPlay` or `ODTSequence`.

The response displayed at the requesting terminal is as follows:

If a SITUation returns FALSE:

SUPERVISOR/<SITUation>:RETURNED FALSE

If a SITUations is TRUE, a response is given of the form:

SUPERVISOR/<SITUation>:RETURNED TRUE FOR <list>

The SITUation must be of a type that can be evaluated statically. For example, a (MX=WAITING) SITUation can examine the WAITING entries, but a (COMPLETED) SITUation merely traps the event of a completion and thus cannot be EVALuated. Similarly, (TAPELABEL) just traps the event of a tape label creation.

All LOG based object classes, such as LOG, and LOGEOJ, as well as MSG, can be evaluated by scanning back through the SUMLOG. To look back at the history of COMPLETED events, LOGEOJ can be used. Similarly, both TAPEDB and TAPELABEL contexts can be used to examine tape entries in the TRIM database.

The '& DEBUG' option gives a printed code trace of the execution of the OPAL program(s).

`<file or directory>` is only required for PD contexts and specifies the file or directory to be scanned.

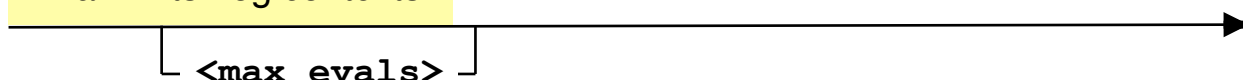
<flex options> is only valid for PD contexts when a valid Flex license exists. The flex options available are Flat,Lin or Fast.

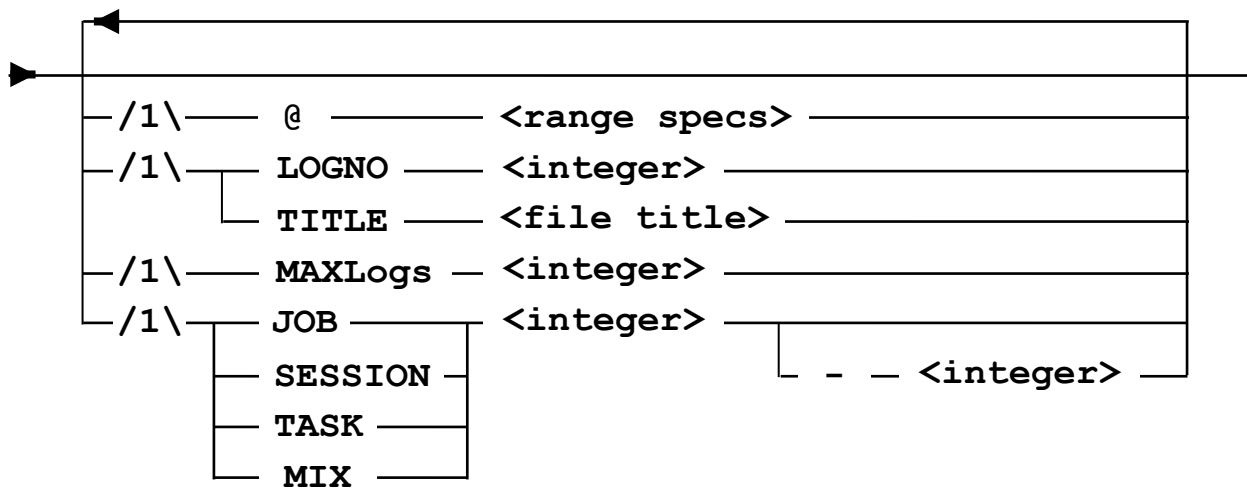
<family name> is only allowed for the SHOWOPEN context. It is used to determine which Family to scan. If omitted the 'System Family' will be scanned. The 'System Family' is specified in the TT USE command. An optional family index can be specified using the # <fam index> syntax.

<Eval limits>

Varies depending on context.

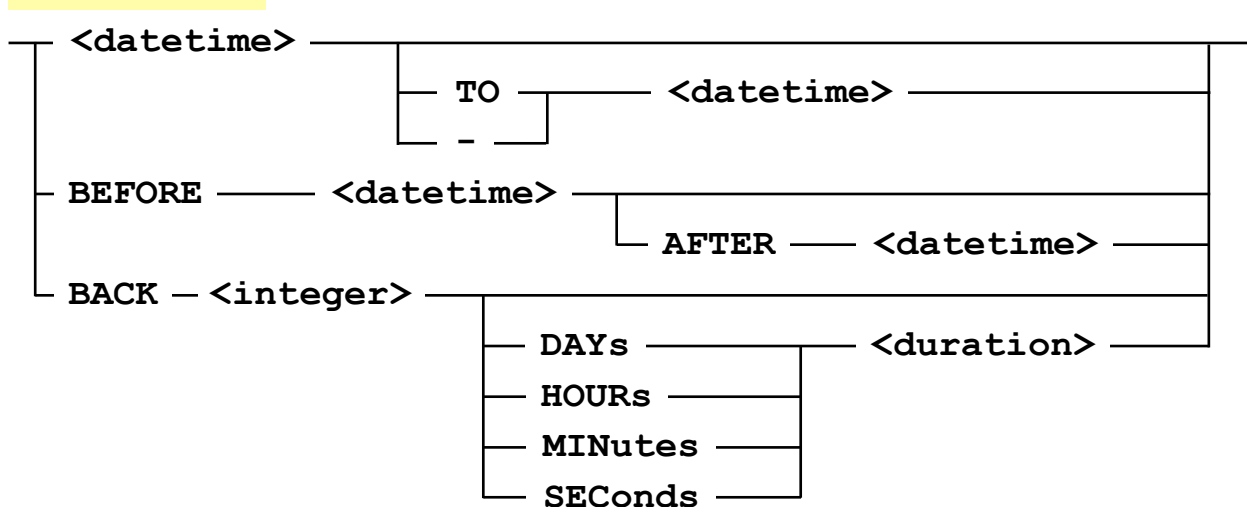
<Eval limits Log contexts>





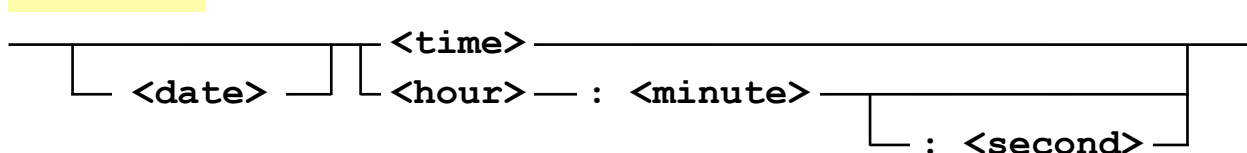
<eval limits> serve to impose restrictions on the evaluation of the SITUation.

<range specs>



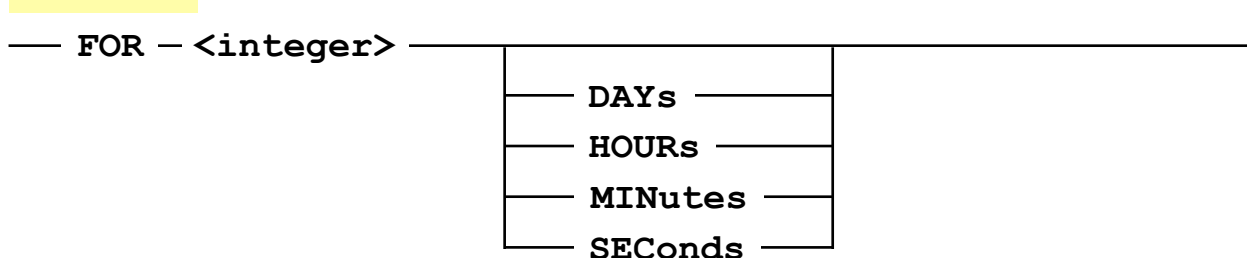
Default units for BACK are seconds.

<datetime>



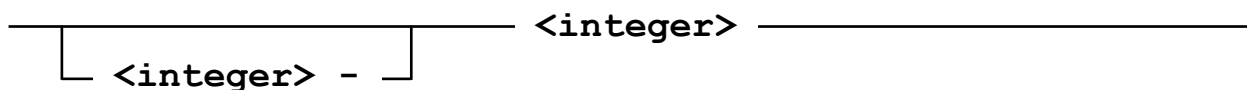
<date> has the same syntax as in the After command., i.e. DD/MM/YYYY, or MM/DD/YYYY if the SO option USDATES is set. <time> also is the same as in After, i.e. an HHMM military format. <hour>, <minute>, and <second> are all 1 or 2 digit integers. If no date is specified and the time specified is greater than current time, yesterday is assumed; otherwise today is assumed

<duration>



Default is seconds.

<max evals>



Enables the User to specify the maximum number of times the Situation will be evaluated before it is terminated and optionally that some evals should be skipped

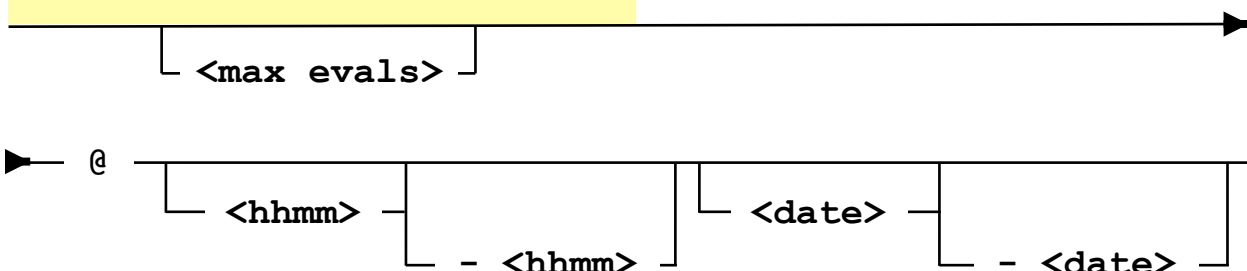
[5-10] indicates that the first 4 objects should be ignored and that the eval should stop after 10 objects. (4 skipped + 6 evals)

[0] is a special case which may be used to test the validity of an Eval command.

If no threshold is specified, the value 10,000 is taken as the default unless any other SUMLOG search parameters have been specified. These can be date or time controls or the specification of a log number or a range of logs and are discussed further below.

In these cases the <max evals> limit will not be assigned unless a BEFORE modifier has been specified without any accompanying AFTER condition.

<Eval limits After context>



If <Eval limits> is not used then the general daily schedule (or parts of the schedule if a subcontext is specified), will be processed. However, using <Eval limits> causes this behaviour to change if a specific date or date range is specified. This forces SUPERVISOR to return all activities that are scheduled to be executed on all days in that range. In particular, activities that have EXCEPT HOLIDAYS or ON WORKDAYS or ON DATE will only be displayed if they are to be processed on that day. The list of activities will be identical to that seen using the command AF ? ON DATE.

```
EV (AFTER:TRUE) [@1000-1300 20/2/08] DO (SHOW(ADDTEXT))
EV AFSITU [@21/02/08-23/02/08] DO AFODTS
```

Using a time range without a date or date range will cause the general schedule to be filtered by time only. In the event a date is specified, the SCHEDULEDAY and SCHEDULEDATE attributes will be automatically set to the appropriate date.

<Eval limits others>

—— <max evals> ————

Specifying a maximum evaluation limit of zero for any EVAL or WHEN command will now cause that EVAL or WHEN to be checked for syntax only and will not be invoked. This has special relevance for "in-line" OPAL commands passed to the TT function, allowing them to be verified before use.

For example:

```
$A:= TT("EVAL " & $SITUEXP & " [0] DO LOGOPAL");
```

If the above statement is accepted by SUPERVISOR, compiling the in-line SITUation specified in \$SITUEXP, then "OK" will be returned to TT (and therefore stored in \$A). The TTLASTERROR flag will be also set to ON if EVAL has failed.

<max entries>

——— <integer> ————
└── <integer> - ──┘

Enables the User to specify the maximum number of times that ODTSequence or DISPLAY can invoked before the WHEN is terminated and optionally that some entries should be skipped.

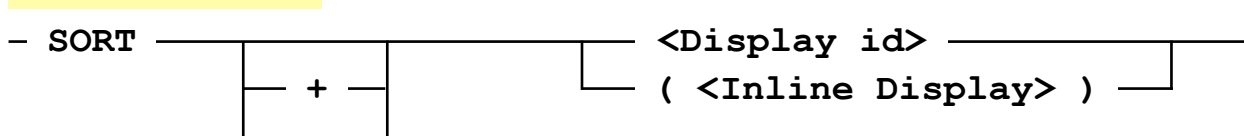
The remaining options are relevant only for SITUations, which are evaluated by reading SUMLOGs. The object classes involved include MSG, LOG, FILEOPEN etc. SUPERVISOR uses a highly efficient log reading procedure, which reads the SUMLOG backwards from most recent time first to oldest last. It will also automatically skip back to previous SUMLOG files.

MAXLOGS is used to limit the number of SUMLOGs that SUPERVISOR will read back through. LOGNO and TITLE allow the EVALuate to start from the specified log file. When TITLE is specified, anything can be in the title before SUMLOG/, but all prior logs must have the same name prefix. The default starting log file is *SYSTEM/SUMLOG. LOGNO specifies a <log sequence number>. The previous logs are automatically found if they are on the DL LOG family, or failing that on the TT USE FAMILY FOR LOGS. Once the family has been changed to the TT USE family, this EVALuate will no longer use the DL family. To be found automatically log names must end in SUMLOG/<systemserial number>/<mmddyy>/<log sequence number>. When switching back from *SYSTEM/SUMLOG, the name of the prior log must be *SUMLOG/...etc.

The "@" syntax is used to limit the log search by time. SUPERVISOR starts at the BEFORE time, which defaults to the latest time in the log file, and reads back until the AFTER time, which defaults to the infinite past. The TO or "--"

syntax, is a more concise way to specify both limits, which can be given in any order. Restricting the search by time can significantly speed up the command because SUPERVISOR is able to binary search to the appropriate start point, and can recognise when the end point is reached.

<Sort Clause>



The objects found by a Situation in an EV command can now be sorted before being passed to the Display or ODTSequence. A <Sort Clause> is added to the command.

Examples:

```

EV ALLMX DISP (MIXNO,,TIME(PROCTIME)) SORT - (PROCTIME)
EV (PD:TRUE) DISP (SEGMENTS 8,,TITLE) [24] SORT (SEGMENTS) (*)= ON =:FL
  
```

SORT - means "sort descending", SORT + "sort ascending". If neither "+" or "-" is specified, it defaults to 'descending'. By using the new OPAL function, COMPLEMENT, multiple keys can be sorted in different directions. The Display given in the <Sort Clause> returns a String, which is used as a key for the sort.

There is a slight difference in the way the Display is evaluated, compared with normal use. In a Sort, an <OPAL String> element has a fixed width. If none is specified, then for a String, it is assumed to be 64, and for an arithmetic or mnemonic, it is assumed to be 12. The Display may not contain a new line (/) delimiter or have a key length of more than 512 chars. Literal strings and the ",," delimiter are allowed, but are unlikely to be useful.

If an entry limit is specified, this is only applied after the Sort, so all of the objects will be passed to the Situation. However, to keep the overheads in control, if no entry limit is specified then an entry limit of 256 is imposed.

Linking a SITUation to a DISPlay

An EValuate does not have to link the SITUation, in which case the only effect is to report the number of objects found back to the requesting terminal. Normally however, the objects found by the SITUation are passed on to another OPAL program. To pass the objects to a DISPlay, the DISPlay or PPrint options are used, followed by an <OPAL def>, which is the name of an existing DISPlay, or an in-line definition. If the in-line definition has no <context>, the object class of the SITUation is assumed. With the DISPlay option, the result is returned

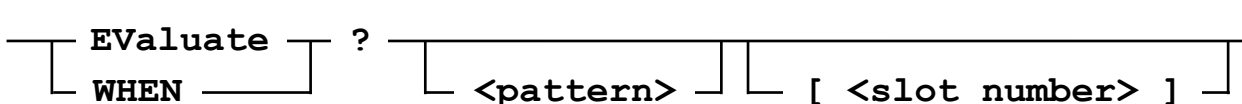
terminal can be changed using the "TT VIA" modifier.

If the PR option is used, all of the matching SITUations will be stopped, irrespective of origin.

The EV command will stop SITUations which have been started by EVal or WHEN.

So EV BOB DO is equivalent to WHEN BOB DO

Interrogating Running OPAL Programs



The EV ? or WHEN ? command produces a display on the requesting terminal of the current status of all current WHENs, ONCEs, DOs, EValuates, and <DISPlay invocation>s.

If an identifier appears, each OPAL currently active will be examined and a response generated if and only if either the name of the SITUation, ODTSequence or DISPlay matches that of the identifier.

If a <pattern> containing wildcards appears, a match is sought.

A typical response to EV ? might look like this:

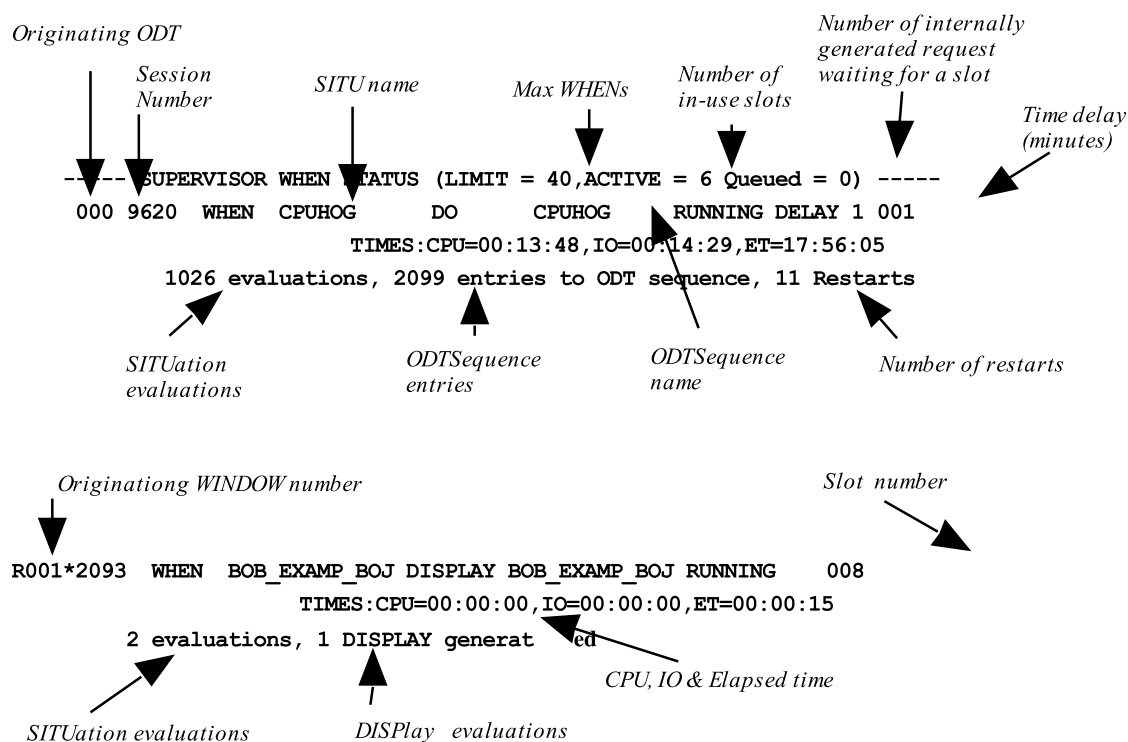
```
EV ?
----- SUPERVISOR WHEN STATUS (Limit = 40, Active = 25, Queued = 0) -----
W 000 18704 WHEN OBI_RESURRECT DO OBI_RESURRECT RUNNING DELAY 1 01
      28608 evals, 117 events, 2233 ODTs entries, 37 Restarts
      TIMES:CPU=00:00:23,IO=00:00:00,ET=18:36:19
W 000 18703 DO MENU_FAMCHK (WAIT TO 09:29.59) 02
      0 evals, 1 ODTs entry, 14 Restarts
      TIMES:CPU=00:00:01,IO=00:00:00,ET=18:36:19
W 000 18702 WHEN OVERRIDE DO OVERRIDE RUNNING 03
      69 evals, 576 Restarts
      TIMES:CPU=00:00:00,IO=00:00:00,ET=18:36:19
W 000 18701 WHEN LOADCD DO LOADCD RUNNING 04
      69 evals, 15 Restarts
      TIMES:CPU=00:00:00,IO=00:00:00,ET=18:36:19
W 000 18700 WHEN RSVP_HANDLE DO RSVP_HANDLE RUNNING 05
      69 evals, 992 Restarts
      TIMES:CPU=00:00:00,IO=00:00:00,ET=18:36:19
001 18699 WHEN META_MAILLBL DO META_MAILLBL RUNNING 06
      4 evals, 1875 Restarts
      TIMES:CPU=00:00:00,IO=00:00:00,ET=18:36:19
W 000 18698 WHEN REC_SEC DO REC_SEC RUNNING 07
      1 eval, 156 Restarts
      TIMES:CPU=00:00:00,IO=00:00:00,ET=18:36:20
```

With a pattern:

```
EV ? META=
----- SUPERVISOR WHEN STATUS (Limit = 40, Active = 25, Queued = 0) -----
001 18699 WHEN META_MAILLBL DO META_MAILLBL RUNNING 06
      4 evals, 1875 Restarts
      TIMES:CPU=00:00:00,IO=00:00:00,ET=18:39:11
W 000 18695 WHEN META_LOGOFF DO META_LOGOFF RUNNING 10
      72 evals, 4 ODTs entries, 156 Restarts
      TIMES:CPU=00:00:00,IO=00:00:00,ET=18:39:11
W 000 18686 WHEN META_SLOTCHECK DO META_SLOTCHECK RUNNING 19
      135 evals, 45 Restarts
      TIMES:CPU=00:00:00,IO=00:00:00,ET=18:39:12
W 000 18681 WHEN META_MAGMAINT DO META_MAGMAINT RUNNING 24
      FOR META TIMES:CPU=00:00:00,IO=00:00:00,ET=18:39:12
      289 evals, 128 Restarts
```

Interrogating the status of active OPAL programs using the EV ? command can be very informative, giving an idea of how busy SUPERVISOR has been. In particular, the numbers of SITUation evaluations and ODTSequence entries linked to the accumulated CPU and I/O times indicate the efficiency and overhead associated with each active OPAL program.

A typical WHEN entry has the following interpretation:



If a slot was initiated from the schedule (i.e. using the AFTER command), The first two columns are replaced by the word "After".

Additional information is returned for a NAPLOG WHEN. Any assigned URC filters previously allocated by the specification of NAPFILTER configuration items will be shown. Where possible, the filter will be depicted as numeric ranges.

It should be noted that all WHENs, ONCEs, and DOs are restarted when SUPERVISOR initialises, unless the NOWHENRESTART option is set. However, if a command did not come from a system ODT, and contains a DISPlay, it will not be restarted.

OPAL programs may appear in a number of states whilst they are active. Usually they will show as RUNNING, but also may appear as INACTIVE (either the WHEN is just being activated or is terminating), QUEUED (waiting to be fully established in a slot) or WAIT-ing. In the latter case, the program might be waiting on a variety of conditions: for example, an EOJ, a message, time or a TT OK <ODTSequence>. The specific condition is shown as part of the WAIT-ing status message.

Other indicators may appear in the response. DEFINE * allows the redefinition of active Situations or routines linked to Situations. If DEFINE * is used then the new code will be used on the next evaluation for Situations, or Entry for other routines. Until the new code has been loaded, a prefix of 'a' before the name of any routine where the code will be replaced. This command allows the code of an active event based SITU/ODTS to be replaced without having to stop and start the routines which could cause some events to be missed. When the new code is actually loaded (on the next Eval or Enter) the 'a' will be removed and a 'code changed' entry will be added to the summary line.

Interrogation by <slot number> can be used to return additional information if the OPAL has a running ODTs, which has been invoked indirectly via a CALL...DO. A <slot number> is an integer, which designates a particular OPAL by its running number.

In this case the ODTs calling sequence will also be displayed. The slot number is given in the message displayed when an OPAL is started, and is also visible in an EV? response.

WHEN states

The different WHEN states are discussed in the following table:

WHEN STATE	DESCRIPTION
RUNNING	WHEN is normal, waiting for or processing events
QUEUED	A new DO or WHEN is in the process of being invoked. This should be a transitory state
WAIT KEYIN	Waiting for a system response to the OPAL KEYIN function via DCKEYIN.
WAIT NAP	Waiting for OPAL attribute information to be retrieved from NAP system libraries
WAIT TLREC	Waiting for Tape Library attribute information to be returned e.g. TL or VIA(TAPE.. requests. Only seen on TRIM slave systems.
WAIT MAIL	Waiting for the completion of an OPAL MAIL function request.
WAIT COMS	Waiting for a response to the OPAL COMS function from SYSTEM/COMS

"FOR <usercode>" clause appears, only the DEFINES locked under the <usercode> will be printed. See the SAVE and DEF ? commands for hints on how to make full use of these features.

OK Command

```

—— OK ——
      |
      |——> <ODTSequence id>
      |
      |——> # <slot no>

```

OK is used to enter an acknowledgement that the operator is ready to proceed with an ODTSequence which has executed an OPAL WAIT(...,OK) statement.

The command requires the OK to be followed by the name of the waiting ODTSequence – note that OK will action all waiting ODTSequences with that name.

To acknowledge a specific waiting ODTSequence, the # <slot no> modifier can be used, where <slot no> is the number of the WHEN slot currently being used.

Examples:

```

TT DEF + ODTs KILLCANDE:
    WAIT("OK TO TERMINATE CANDE?", OK);
    ODT(#[MX=*SYSTEM/CANDE], "SM QUIT CANDE");

```

A DO of the above ODTSEQUENCE, running in slot 22, shows the following state in a WHEN ? response

```

----- SUPERVISOR WHEN STATUS (Limit = 40, Active = 22, Queued = 0) -----
W 223*07816                                DO  KILLCANDE      (NEEDS OK)                22
                                           TIMES:CPU=00:00:00,IO=00:00:00,ET=00:00:03

    0 evals, 1 ODTs entry

```

The DO can be OK-ed using either of the following:

```

TT OK KILLCANDE
TT OK #22

```

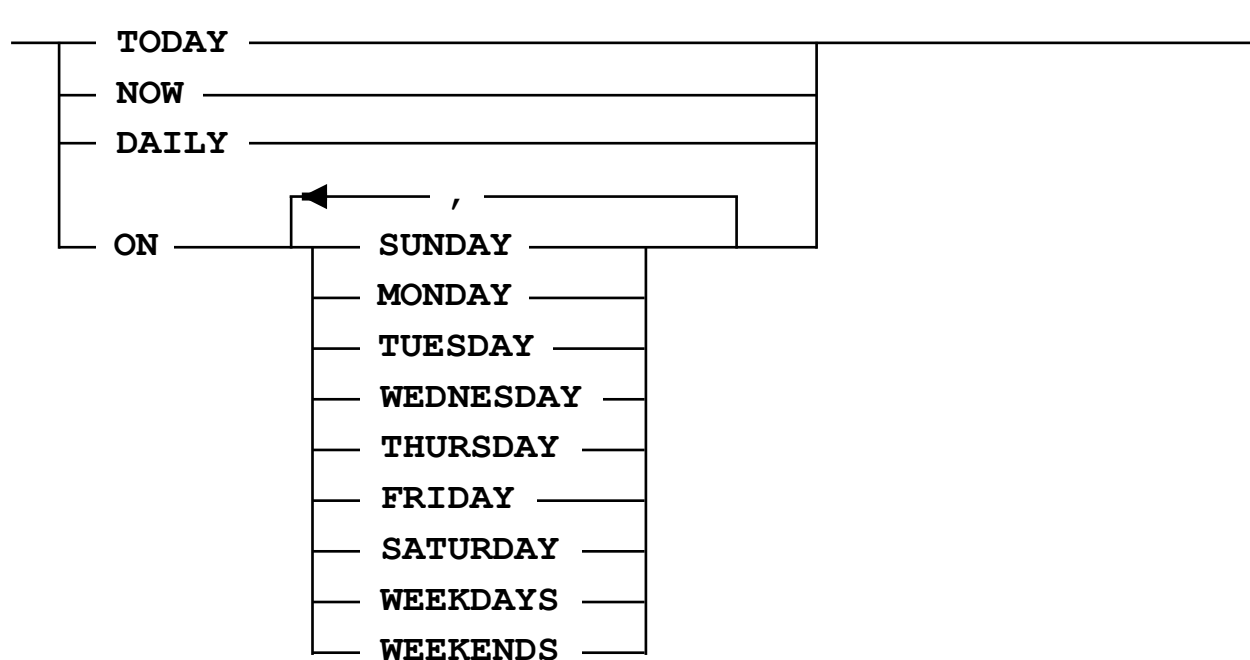
PRINT Command/Modifier.

```

—— Print ——
      |
      |——> ATtributes
      |
      |——> <Display def>
      |   |
      |   |——> & DEBUG
      |   |
      |   |——> <parameters>
      |
      |——> <Printable Report Command>
      |
      |——> SCHEDULE
      |   |
      |   |——> <period spec>
      |   |
      |   |——> BRIEF
      |
      |——> <program type>
      |   |
      |   |——> DEFine
      |   |
      |   |——> <pattern>

```


<period spec>



PRint may be used as a modifier to some SUPERVISOR commands (shown in the railroad diagram as <Printable Report command>s). These commands are currently:

HELP, FP, LU, HO, PDT, TP
MAIL, REC, AF, INPUT, ENTER, SLOT, LOG

After a PRint Modifier, the responses generated by the command will produce a report to the print destination specified by USE DESTINATION.

It is sometimes convenient to get information in hardcopy form on a line printer. The PRint command allows listings of one or many OPAL programs to be sent to a printer. The type of program (DISP, ODTs, or SITU) is specified. The DEF qualifier means print irrespective of type. If no <pattern> is given, all programs of the requested type(s) will be printed.

The PRINT ATTRIBUTES form lists out an alphabetical list of all the ATTRIBUTES that are currently available to an OPAL programmer, along with a concise description of their meanings and use.

The file attributes of the printer file are determined by an optional dummy label equation of LINE bound into the codefile to suit each installation's needs. File attributes that may be changed are MAXRECSIZE (specified in characters per line), PAGESIZE, TRAINID and FORMID. If the value of TRAINID is EBCDIC96, both upper and lower case characters are used.

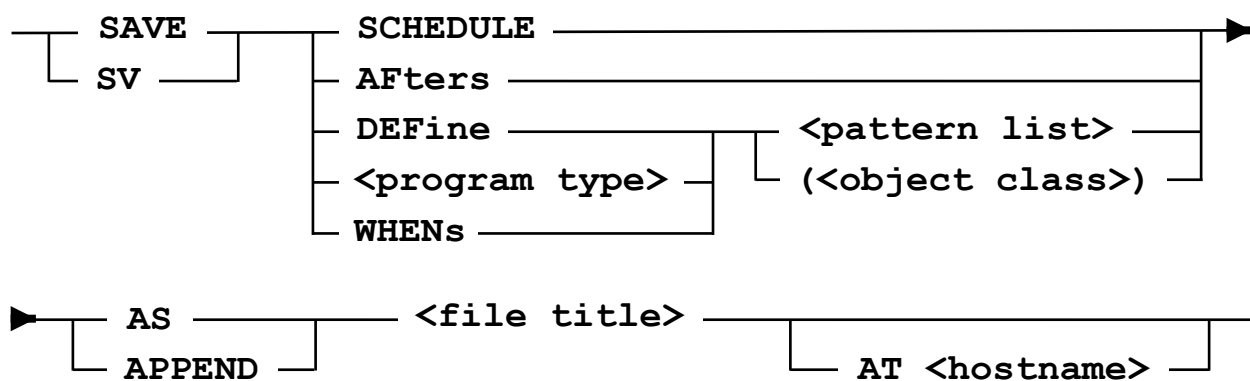
The PRINT <OPAL def> variant allows a DISPlay to be executed (instead of using the "/" command) and have the output sent to the printer, not the requesting terminal. This is analogous to the PRint variants of the EV and WHEN commands. The <OPAL def> is an identifier designating an existing DISPlay or an in-line DISPlay definition. <parameters> are those required for a non-SYSTEM DISPlay. See the DISPlay Invocation (/) command for more details.

The PRINT SCHEDULE variant of the PPrint command produces a report showing scheduled activities for one or more days. By default, if no ON modifier appears, TODAY is assumed. By adding the modifier NOW, the report will only contain the activities remaining from the current time until midnight. The AFTER HL for TODAY will still be included in the output. By adding the optional modifier BRIEF, the normal two-line per entry report is shortened by omitting the second line of time stamps.

Examples

```
TT PR ATT
TT PR SCHED DAILY BRIEF
TT PRINT DEF TAPEL=
TT PR (PER:UNITNO 3,,RF 3,,IOS 8) MT 18-23
```

SAVE Command



The SAVE command is used to implement a multi-level approach to error recovery. There are two forms of the SAVE command: the form discussed here is used to selectively backup the SCHEDULE file and OPAL programs in symbolic format; the other form, described in [Administration Commands](#), is for backing up the SCHEDULE file and the USERDATA file in their binary formats

SUPERVISOR holds information about options, and the sources for OPAL programs, in its own internal file. The contents of this file can be wholly, or selectively written to a JOBSYMBOL file using the SAVE command. It is written in the same format as is needed for an ENTER command, so it can be used as an effective backup method for SUPERVISOR.

SAVE Programs

OPAL programs can be individually or collectively saved by <Program Type>. If <pattern list> contains no wildcards and specifies a single program, only that one program will be saved. If the <pattern list> contains wildcards, any program of the correct type or WHEN command which matches the <pattern list> will be saved. If DEF is used for the type, any program that matches the name will be saved, regardless of type. The programs saved can also be restricted to a specified object class.

Multiple selections of program identifiers to be loaded to the same OPAL symbolic file can be specified using <pattern list>. Each program specified in such lists may be wild-carded. For example:

```
TT SAVE ODTs OPAL=, META_ = AS MERGED/OPALS
```

Only ODTSequences whose names start with OPAL and META_ will be SAVED to the file MERGED/OPALS. Further, the SAVE command allows multiple save actions to be performed to the same file. This is achieved by using the 'APPEND' modifier instead of 'AS'. For example:

```
TT SAVE SITU IPP=, BOB= APPEND MERGED/OPALS
```

Here, any SITUations whose names begin with IPP and BOB will be added to the file MERGED/OPALS. The output file must be of type JOBSYMBOL. Because this process conceivably might allow the production of very large output files, the minimum sequence increment has been set to 100 instead of 1000.

Currently, both SAVE and ENTER will allow up to 24 identifiers to be used in a program list. Attempts to exceed this limit will return the error message:

```
TOO MANY ID'S SPECIFIED IN SAVE/ENTER LIST
```

The use of the (<object class>) syntax allows OPAL programs to be selected by object class. Although syntactically allowed, it does not have any effect on SAVE MEMO or SAVE WHEN.

```
TT SAVE SITU (MX)
```

SAVE SCHEDULE

The most secure, but most costly form of backup is given by the SAVE SCHEDULE AS <file title> syntax. It takes the entire current state of SUPERVISOR – that is all the options and programs – decompiles them, and puts them on the JOBSYMBOL file specified by "AS <file title>". This file can be edited by CANDE if necessary, and can be re-input to SUPERVISOR by an ENTER command.

In a SAVE SCHEDULE AS command, if the <file title> is

```
(<USE_USER>) SAVED/SCHEDULE/DEFAULTS ON <backup family>
```

This file is a potential emergency backup which SUPERVISOR will automatically try to find should it be unable to find a valid SCHEDULE on the DL JOBS family or a backup copy on the BACKUP family with the title *SAVED/SCHEDULE. If neither of these files can be found, SUPERVISOR will create an empty SCHEDULE and perform an automatic ENTER on the saved default file before proceeding.

The other variants of the SAVE AS syntax allow decompilation by category, so, for example, SAVE ODTs will create a file containing the text of all ODTSequences.

The <pattern> modifier is used to select a group of programs to be saved. A <pattern> may refer to a single WHEN, DEFINE or group of programs, or all of the programs of a given type, where type is one of ODTs, SITU, DISP, or MEMO. If the

type is "DEFINE", then all of the above types will be referenced. If a <pattern> appears, each WHEN slot, or each DEFINE in the OPAL program dictionary of the proper type, will be examined and its name will be compared to the <pattern>. If a match occurs, the corresponding definition will be saved.

If a "FOR <usercode>" clause appears before the SAVE, only the WHEN or DEFINES locked under that <usercode> will be saved.

The partial save of definitions, SITU, ODTs, DISP and MEMO allows transfer of new definitions across machines without having to edit the symbolic file of scheduled activities.

These saved symbol files are particularly useful on BNA networks, as they can be transferred by a simple copy, or by an ENTER command specifying the file title to which definitions were saved, and the HOSTNAME on which that file is resident.

TT SAVE SCHEDULE AS LX100A/SCHEDULE AT LX100B

If appropriate naming conventions are chosen to group related DEFINES, locating, using, and maintaining these DEFINES can be quite simple. For example, prefixing all of the DEFINES relating to CANDE with "CANDE_" makes it easier for the operator to locate these, e.g. CANDE_RUN, CANDE_SETUP, CANDE_MEMO, CANDE_SM, etc. Note that these will also appear together in a DEFINE ? command.

SAVE AFTERS

This command allows all scheduled activities (i.e. AFTER commands) held in the current SCHEDULE file to be saved to its own symbolic file. This information is also decompiled by the usual SAVE SCHEDULE AS.. command but this variant is much faster and convenient.

In addition to the above change, the handling of AFTERS has been improved; any AFTER command will now have multiple spaces removed from the command text, unless the spaces are in a quoted string.

This permits AF commands to be more easily deleted (if using the AF-..<text> variant) without having to ensure that the original number of spaces is preserved. For existing AF commands already held in the SCHEDULE, SUPERVISOR will automatically deblank when compared with AFTER commands for addition or deletion.

SAVE WHENS

The SAVE WHENS variant allows all active WHEN and DO slots to be written into its own symbolic file. This information is also decompiled by the usual SAVE SCHEDULE AS.. command but this variant is much faster and more convenient.

Examples

```
TT SAVE ODTS AS (OPERATIONS)CURRENT/ODTS ON OPS
TT SV DEFINES COMS= AS (OPS)DEFS/COMS ON DISK
TT SAVE WHENS AS (META)SAVED/WHENS ON DEV
TT SAVE SCHEDULE AS DECOMPILED/SCHEDULE
TT SV AFTERS AS ALL/AFTERS AT LX100A
TT SAVE DEF IPP/=, BOB/= APPEND META/OPALS
```

SAVED SCHEDULE FILE SAMPLE LISTING

```
SO + 29,22,10,7\  
SO - 47,46,45,44,43,42,41,40,39,38,37,36,35,34,33,32\  
SO - 31,30,28,27,26,25,24,23,21,20,19,18,17,16,15,14\  
SO - 13,12,11,9,8,6,5,4,3,2,1,0\  
USE QUEUE 255\  
USE ODT 2\  
FOR SUPERVISOR USE USER SUPERVISOR\  
FOR SUPERVISOR USE USER TAPELIB FOR TAPELIB\  
FOR SUPERVISOR USE USER EXTRN FOR EXTERNAL\  
FOR SUPERVISOR USE USER T6YW8 FOR ODTSECURITY\  
USE LANGUAGE 0\  
USE FAMILY DISK FOR SYSTEM\  
USE FAMILY LOGPACK FOR LOGS\  
USE FAMILY LOGPACK FOR SAVES\  
USE FAMILY WRKPACK FOR EXTERNAL\  
USE JOB *TL/HANDLER AFTER TL\  
USE TASK *TEST/REC FOR RECORDER\  
USE DESTINATION SUPERVISOR\  
HOLIDAY 25/12\  
HOLIDAY 01/01\  
HOLIDAY 13/11/98\  
  FOR HELP                                DEFINE + SITUATION BADPU (MX=BOT) :  
    USERDATA (PU) AND USER NEQ "HELP"  
\  
  FOR HELP                                DEFINE + ODTSEQUENCE DSIT:  
    ODT (MIXNO, "LP-") ;  
    ODT (MIXNO, "DS") ;  
\  
DEFINE + SITUATION RSVPCHECK (MX=W) :  
  TRUE  
\  
DEFINE + ODTSEQUENCE RSVPCHECK (MX) :  
  RECORD [RSVP] (MIXNO, ,RSVP) ;  
\  
AFTER + 0000 ON SUNDAY TT DO TEST\  
AFTER + 0110 ON SUNDAY START BATCH/JOB ON DEV\  
VIA 1 FOR HELP WHEN BADPU DO DSIT\  
DO ASMN_BASIC (FLEX) DUMPDEV\  
WHEN RSVPCHECK DO RSVPCHECK
```

See Also :

[DEFINE ? Command](#)

[ENTER Command](#)

[USE command](#)

SLOT Command

SLOT <slot number>

The SLOT command returns an in-depth analysis of the specified WHEN environment. The response shows current state information and values for the variable heap. The PRINT modifier can be used with this command. As well as sending the output to a print file it enables the dumping of more detailed slot environment information and local array data.

The '+' modifier suppresses the dumping of the variable heap and local array data but additionally displays detailed slot environment info.

Find the slot we are interested in.

```

ev ? meta=
----- SUPERVISOR WHEN STATUS (Limit = 40, Active = 25, Queued = 0) -----
001 15042 WHEN META_MAILLBL DO META_MAILLBL RUNNING 06
TIMES:CPU=00:00:00,IO=00:00:00,ET=166:08:02
32 evals, 1873 Restarts
W 000 15038 WHEN META_LOGOFF DO META_LOGOFF RUNNING 10
TIMES:CPU=00:00:03,IO=00:00:00,ET=166:08:02
713 evals, 28 ODTs entries, 154 Restarts
W 000 15029 WHEN META_SLOTCHECK DO META_SLOTCHECK RUNNING 19
TIMES:CPU=00:00:00,IO=00:00:00,ET=166:08:05
990 evals, 43 Restarts
W 000 15025 WHEN META_MAGMAINT DO META_MAGMAINT RUNNING 24
FOR META TIMES:CPU=00:00:01,IO=00:00:00,ET=166:08:05
1347 evals, 126 Restarts

```

Display analysis of WHEN META_MAILLBL DO META_MAILLBL (Slot 06)

```
slot 6
---- OPAL DUMP FOR SLOT #6 At 08:14:21 ---
Entered by : WHEN META_MAILLBL DO META_MAILLBL
Handled 32 Evals, 0 ODTIS/DISP entries,1873 Restarts
Time since last SITU event: 4:57:48.90
Times: CPU= 0:00:00.05, IO= 0:00:00.03, Elapsed= 166:08:51.43
Originated from ODT 1 Session 15042

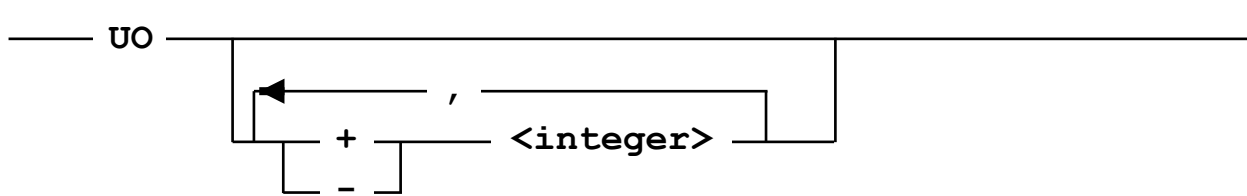
--- OPAL STACK INFO ---
OPAL Version=5252060, MyTI=16
MySITUATIONTYPE = 0A000000000000    MyAWAITS          = 0000000008000
MyWAITCLASS      = 0000000008000
Not executing code, waiting on event
TOPOFSTACK = 10
STAK[10]    = 0000000000000
STRINGTOS   = 0
STR2B is STRINGS[0,*], STR is STRINGS[0,*]

-- STRINGS [0] 6 bytes --
00000000:PD0021

--- Local heap not initialised ---
```

If a slot has been initiated from the schedule (i.e. via an AFTER command) then 'Originated from Schedule' is displayed instead of an ODT or Window.

User Options (UO) Command



The value of <integer> must lie between 0 to 47 inclusive. UO + sets the flags numbered in the list following, and UO - resets them. SUPERVISOR takes no action if user options are set or reset; their meaning is assigned by OPAL programmers who can access the option settings with the UO attribute. The settings of user options are preserved across halt loads or restarts of SUPERVISOR.

Examples

```
TT UO +1,2,3, -4,5,6
```

UO shows which flags are currently set, giving a response of the following form:

```
TT UO
USER OPTIONS CURRENTLY SET:NONE
```

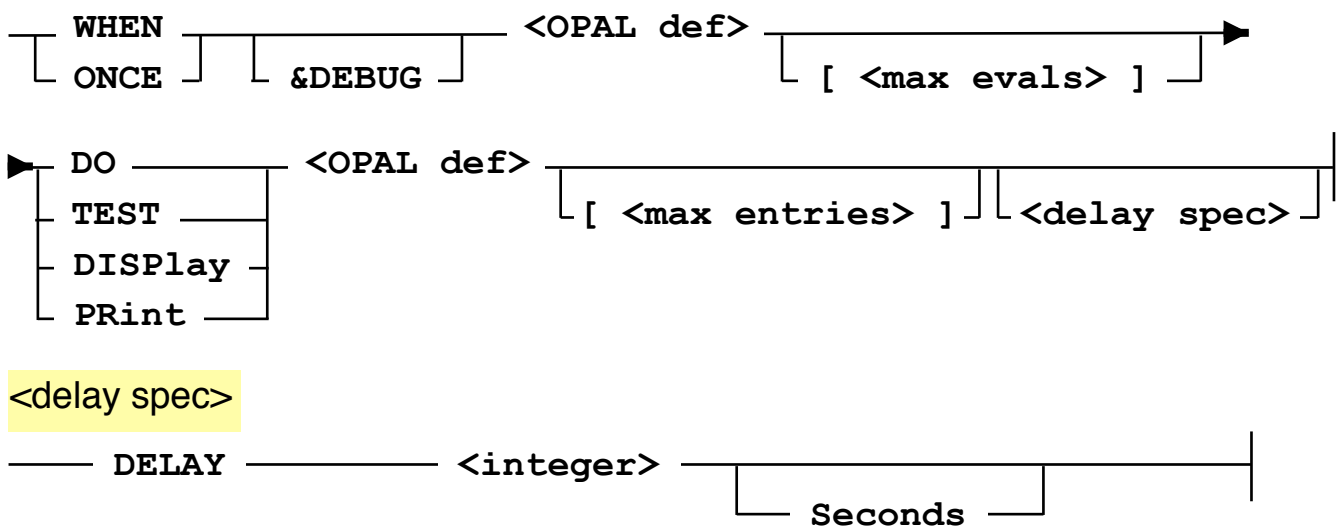
Or

```
TT UO
USER OPTIONS CURRENTLY SET:0,5,8,23
```

WHEN Command

The WHEN and ONCE commands control monitoring SITUations, that is OPAL programs called SITUations, which are run continuously.

Monitoring with an OPAL SITUation



<max evals> and <max entries> are integers.

Specifying a maximum evaluation limit of zero for any EVAL or WHEN command will cause that EVAL or WHEN to be checked for syntax only and it will not be invoked. This has special relevance for "in-line" OPAL commands passed to the TT function, allowing them to be verified before use.

There are two programs specified in the WHEN command, a SITUation and either a DISPlay or an ODTSequence. SUPERVISOR allocates a "slot" in the OPAL Virtual Machine and puts out a message such as:

```
SUPERVISOR <id>+<id>: BEGIN MONITORING (SLOT 3)
```

There are some options in the syntax of the WHEN command to run programs.

ONCE

For SITUations activated by the WHEN command, whenever the SITUation occurs (evaluates TRUE), the corresponding action is taken. The WHEN continues until it is deactivated. ONCE differs from WHEN because it stops after the first time the action is performed. It is thus equivalent to a WHEN with **<max entries>** of 1.

& DEBUG

Is used to debug problematic OPAL programs. It causes a trace of the execution within the OPAL machine to be written to a printer file. This trace is also produced automatically if a program has a fault.

DISP

Fire off a DISPlay to run on the originating terminal whenever the SITUation occurs.

PR

Fire off a DISPlay to send output to a printer file whenever the SITUation occurs. The WHEN must be terminated to see the print out automatically.

DO

Executes an ODTSequence when the SITUation occurs. Note that a SITUation, which is running with an ODTSequence, may not be run with any other ODTSequence or DISPlay.

TEST

TEST is identical to DO, with an important exception. All statements that have an action are turned to DISPlays. This means that an ODTSequence can be debugged safely before being run and no special changes or debugging code are needed.

DELAY

The DELAY clause controls the frequency of evaluation of time-based SITUations. SECONDS (minimum abbreviation "S") can appear after the integer DELAY to denote that the delay is in seconds. The maximum delay is 4095 seconds. The default DELAY is 1 minute.

For example:

```
TT ONCE LOW_DU [100] DO LOW_DU DELAY 90 S
```

The NOTCOPIED_MSG OPALs allow messages that indicate files that were NOT copied during a LIBRARY/MAINTENANCE copy job to be picked up and converted into waiting entries. In the examples below, the SITUation can pick up such messages very quickly because of the category and message number checks. Also, the VIA function allows us to determine the usercode of the task that was doing the copy. In this case we are only interested in jobs invoked by the FLEX usercode.

The WHEN command needed is:

```
TT WHEN NOTCOPIED_MSG DO NOTCOPIED_MSG
```

where

```
DEFINE + SITUATION NOTCOPIED_MSG (MESSAGE) :  
    MSGCAT=MSRDIR AND MSGNO= 10 AND      % FILE NOT COPIED MSG  
    VIA (MIXNUMBER: USER="FLEX")
```

```
DEFINE + ODTSEQUENCE NOCOPIED_MSG (MESSAGE) :  
    ODT ("BEGIN JOB ", VIA (MIXNUMBER: VIA (JOBNUMBER: NAME)) ,  
        " ; STRING S ; S:=ACCEPT ( " , TEXT , " ) " ) ;
```

See Also

[DEFINE Command](#)

[<DISPlay Invocation> Command](#)

[DO Command](#)

[EValue Command](#)

Activating SITUations in WHEN slots

Both the WHEN and ONCE commands allow a SITUation to be linked to either an ODTSequence or a DISPlay. Entering of one of these commands initiates the following process:

- If the first <OPAL def> is an <OPAL id>, verify the existence of a SITUation definition with the specified name, otherwise compile the inline OPAL.
- If the second <OPAL def> is an <OPAL id>, verify the existence of the ODTSequence or DISPlay with the specified name, otherwise compile the inline OPAL.

- Verify that the type of the SITUation and the type of the ODTSequence or DISPlay are the same. SYSTEM SITUations may only be linked to SYSTEM ODTSequences or DISPlays, and so on.
- If the request is to link a SITUation with an ODTSequence (WHEN...DO variant), verify that the SITUation is not already active (linked to any ODTSequence or DISPlay). If it is already active, deny the request.
- If the request is to link a SITUation with a DISPlay (WHEN...DISPLAY variant), verify that the SITUation was not previously activated from the same source (System Console, REMOTESPO, etc.). If it is already active, deny the request.

At this stage, the availability of a "WHEN slot" is checked. During compilation of SUPERVISOR, an absolute upper limit to the number of simultaneously active SITUations is set. If a slot is available, the request is honoured immediately. If no slot is available, the request will be rejected.

All information necessary to restart is held permanently in the SCHEDULE. Activated SITUations are maintained across Halt Loads.

When a SITUation is activated, a Session is allocated.

The SITUation remains active until explicitly deactivated by means of the WHEN command, or until a fault occurs evaluating the SITUation or executing the ODTSequence or DISPlay. Should a fault occur, SUPERVISOR would terminate the SITUation with a message indicating abnormal termination. A program dump of SUPERVISOR will always be taken in this case.

Should an ODTSequence execute a WHEN Statement as one of its statements the effect is as if the ODTSequence terminated, and a new WHEN command had been entered from the SPO. However, the same "when slot" is used for the new WHEN, thus guaranteeing that the request finds a slot.

How WHENs are Evaluated

As part of the activation process, if a SITUation can be evaluated it will be. If the SITUation has occurred, the ODTSequence or DISPlay is at once executed; SITUations activated with a ONCE command are then deactivated.

After this first evaluation, the SITUation returns control to SUPERVISOR. This Section discusses the events that cause SUPERVISOR to next evaluate a SITUation.

If the SITUation is of object class MX (except MX=BOT), or PER, it will first do an evaluation on the current objects. If the SITUation is time-based it is evaluated once when it begins monitoring. SITUations linked to events will only evaluate when their event is caused. Time-based SITUations will evaluate with the frequency given in the DELAY clause.

Just as there are two types of ADM possible for most ADM displays, there are two

types of SITUations possible. SITUations dependent on ATTRIBUTES, which are constantly changing without signalling an event, logically require SUPERVISOR to periodically poll the values to evaluate the SITUation. Such SITUations should be DEFINEd with the TIME modifier. The default sample period (between polls) is 1 minute. To enable this period to be changed, the DELAY modifier to a WHEN command is provided, and any period up to 4095 seconds may be specified. For example, there is little point in incurring the overhead of checking the values of DU (DUMAX) for all pack families every minute – 5, 10 or even 15 minutes is a reasonable sample period for this type of activity.

For event based SITUations, SUPERVISOR evaluates only when necessary.

There are some ATTRIBUTES of type MX that cannot change during the life of a program. If only these ATTRIBUTES are involved in a SITUation definition, the SITUation need only be evaluated once for each new mix entry. These ATTRIBUTES are described in the report produced by the "PR ATT" command as:

CONSTANT FOR THE DURATION OF THE PROGRAM

The OPAL Compiler expends considerable effort in deciding what status changes a SITUation needs to be notified of, and the information is used by SUPERVISOR in deciding when to awaken a WHEN.

It is important to have the possibility of "event mode" MX and PER SITUations, as there are many transient conditions which can only be detected and acted upon if the MCP notifies SUPERVISOR as they occur. However it is easy, particularly with MX SITUations, to consume more resources within SUPERVISOR than is desirable. The MX class, without a subclass, acts like MX=A,W,S. Often techniques like restricting MX to MX=WAITING or even MX=ACTIVE, or simply changing the order of evaluation so that complicated string expressions appear after simple Boolean or arithmetic expressions, can dramatically reduce the cost of a SITUation.

Normally, event-based SITUations of class MX, which are run in a WHEN, start with an Evaluate. Only after the first evaluation do they become event-based. If SITUations (with the MX event sub-contexts of BOT, GOING, or PRIORITY) are part of a WHEN, they will be evaluated whenever the appropriate event occurs. SUPERVISOR will not search the mix to evaluate these SITUations.

Locking WHENs

It is possible to "lock" WHENs, that is to activate WHENs and DOs with an associated usercode so that they cannot be deactivated without knowledge of a password associated with the usercode. The usercode is obtained in the normal way, either from an explicit FOR modifier or because there is an implied usercode associated with the source from which the command originated. SUPERVISOR's behaviour when activating a locked WHEN is modelled on the MCP's search for files.

To lock a WHEN the SITUation must have been DEFINEd with an associated usercode. The definition is handled exactly as before, except that it is entered in the

SITUation Dictionary with a name constructed from both the name and the supplied identifier.

For example

```
FOR A/B DEFINE SITUATION TESTER(MX) : TRUE
```

would enter a SITUation called **(A) TESTER** into the dictionary.

Definitions of ODTSequences and DISPlays work in the same way. As SUPERVISOR does NOT permit a command like

```
DEF ? SITU (A) TESTER
```

A definition entered in this way can only be referred to by using a FOR statement, in a DEFINE, WHEN, EVALUATE, ONCE, DO, or / command.

If a command like

```
FOR A/B WHEN X DO Y
```

is entered, SUPERVISOR searches first for a SITUation with the name (A)X, secondly for X. Similarly first (A)Y and then Y will be sought. To deactivate the above WHEN, two cases arise. If an operator enters

```
WHEN X DO Y
```

then if the SITUation running is really (A)X the command is disallowed. If the SITUation running is X the command is allowed even if the ODTSequence running were (A)Y.

To "lock" a WHEN, at least the SITUation and preferably the ODTSequence or DISPlay should be DEFINEd under that (same) usercode.

In the case where the operator enters

```
FOR A/B WHEN X DO
```

the WHEN would be deactivated regardless of whether (A)X or X were actually running.

As with the AFTER command, a privileged usercode can overcome his lack of knowledge of a password of a non-privileged usercode by giving two FOR clauses, the first being his usercode and password, the second specifying the other usercode, without password.

[See For](#)

Logging of WHENs and Dos

This section covers logging for resource accounting purposes of WHENs and DOs. Most separate stacks and distinct chargeable activities initiated by SUPERVISOR are assigned a job number different to SUPERVISOR's job number. This is done, firstly, to aid resource accounting, and secondly, so that any printout generated by a stack will be immediately available when the stack completes. Job numbers are obtained from the MCP by means of the MCSLOGGER intrinsic, in the same way

CANDE allocates sessions to its users.

When a session is allocated, it is possible to supply a usercode, chargecode, and accesscode which together "label" a job for resource accounting purposes. The standard job summary printed for jobs and sessions by AUTOPRINT is headed by these three values in large friendly block letters. SUPERVISOR assigns its usercode (set by the USE USER command) to all sessions.

Resource Accounting for Compilations.

Upon entry of a DEFINE command, SUPERVISOR initiates the OPAL Compiler as separate stack. This is done so that any printout generated by the compilation will be immediately available when the compilation completes. SUPERVISOR assigns its usercode (set by the USE USER command) to all sessions. The chargecode of the session is set to the name of the entity being DEFINEd (SITUation, ODTSequence, or DISPlay). The resources used by the OPAL Compiler are charged to that session.

Execution of WHENs and DOs.

Upon entry of a WHEN, DO, or EVALuate command, SUPERVISOR queues a request to initiate a new "WHEN". At the point where the request is de-queued, a session is allocated for the request. This session is allocated with SUPERVISOR's usercode, an accesscode corresponding to the name of the SITUation, and a chargecode corresponding to the name of the ODTSequence or DISPlay. In the case of a DO or an explicit request for a DISPlay, there is no SITUation and the name "DO" is used for the chargecode. In the case of an EVALuate, there is no ODTSequence and the name "EVALUATE" is used for the accesscode.

All resources used within SUPERVISOR, including calls on MCP functions, are charged to this session. The log reason will be "log-on by hello". The current session number of a WHEN is displayed as part of the response to the WHEN ? Syntax.

WHENs and DOs that terminate normally will have their log-off reason set to "normal log-off". Those that terminate abnormally will have a log-off reason of "error abort". Some ODTSequences have the ability to change SITUation and ODTSequence that are linked together (the WHEN command in an ODTSequence). Should this occur, the session is split and a new session is allocated.

The intent of this feature is to enable sites to balance the benefits of having SUPERVISOR monitor for, and respond to occurrence of SITUations against the overhead required.

WHEN and DO Session Recovery

Should SUPERVISOR terminate abnormally, or the system Halt Load, upon its next initiation SUPERVISOR will attempt to release all sessions that it had allocated. In certain cases, this will no longer be possible, for example, after a change of the Halt

Load unit; the MCP may have no knowledge of the session numbers.

When sessions are released in this way, the log-off reason will be given as "restart of MCS". The values of processor time and I/O time used by the session remain valid.

Splitting Sessions over a TL

To avoid having sessions allocated over more than one SUMLOG, SUPERVISOR, prior to releasing the SUMLOG, will release all current sessions in use by WHENs, giving the log-off reason as "session split". After the log release is complete, new sessions with a log-on reason of "split session" will be assigned.

Logging of Tasks

SUPERVISOR initiates dependent tasks and independent jobs. For the purposes of logging and accounting, a separate usercode, which is determined dynamically by each site, is required. Only the name need be specified to SUPERVISOR; the program itself maintains the usercode by means of the USERDATA intrinsic. This is done because the usercode must be privileged.

See the SUPERVISOR Options LOGMINIMAL and MONITORING in [SUPERVISOR Control Commands](#) for details of what is logged by SUPERVISOR.

To avoid unauthorised use, SUPERVISOR generates random passwords for this usercode periodically – usually every four hours. Further all tasks initiated by SUPERVISOR are assigned a unique and descriptive chargecode, the name describing the function of the task.

Dependent tasks initiated by SUPERVISOR are assigned a job or session number in much the same way as CANDE sessions are allocated, by means of the MCSLOGGER intrinsic. This is necessary to allow printer backup and card punch output to be available for printing when the task goes to EOT, rather than when SUPERVISOR itself goes to EOJ.

Logging and the TL Command

SUPERVISOR's TL command is a small, but significant variant of the TL system command, and both perform a release of the current log file. However, it is inconvenient to have sessions allocated for which the log-on entry is in one log and the log-off in another. For this reason, SUPERVISOR splits all WHEN sessions prior to the log release. After the log release, it allocates new sessions.

If any SUPERVISOR initiated tasks are running, the same problem occurs, except that there will also be a BOT log entry with no matching EOT entry in the first SUMLOG. In that case, it is not possible to release a session for an active task. The recommendation is not to enter a TL command when SUPERVISOR has tasks active.

Caching ODTSequence code

In earlier versions of SUPERVISOR only the code required to evaluate a SITUation and execute any associated ODTSequence would be held in memory once the code segments had been loaded from the SCHEDULE file. If an ODTSequence performed a CALL DO, the code for the named ODTSequence would be loaded from the SCHEDULE, executed and then discarded. In circumstances where an ODTSequence was structured in this way the code for the CALLED ODTS would be reloaded each time it was required.

In current software, SUPERVISOR will attempt to store the code for a CALLED ODTS in memory, subject to resource limitations. This code memory limit is currently 25684 bytes, which is used to hold both the original SITUation and ODTSequence code plus any CALLED ODTSequences used within the WHEN slot. Once the cache allocation is exhausted, SUPERVISOR will revert to loading code from the SCHEDULE file as required.

It is also possible to modify an ODTSequence that has already been invoked by a CALL DO - the new code is executed on the next occasion that the ODTSequence is CALLED. When such code is re-DEFINEd, a message will be displayed indicating that the cache for any WHEN slots where the code is currently in-use will be purged.

Scheduling

This chapter deals with the scheduling facility of SUPERVISOR. In a standard A Series operating environment, commands to the system are always executed immediately and exception conditions frequently require attention. This means that operations staffs need to be present at all times. The Scheduler in SUPERVISOR introduces a new possibility; the delaying or scheduling of system commands for a specific time or range of times in the future. This is one of the most powerful and useful features of SUPERVISOR, implemented by the AFTER command. The AFTER command allows an activity can be DEFINED to occur once at a specific time or to be repeated at regular intervals. The syntax for defining time is such that it is both natural for humans and precise enough to pinpoint the time or times intended.

As well as the AFTER command, there is another scheduling-related command, HOLIDAY, which is used to DEFINE those dates when normal working does not take place. Both the AFTER and HOLIDAY command, as well as the time specification syntax, are discussed in this chapter.

Changing the System Time and Date

Changes to the system time and date are **strongly discouraged** since SUPERVISOR can get very confused if the time is set backwards. In general, these problems can only be resolved by not setting the time, except immediately after a HALT LOAD and before entering TIMEOK.

Having said that, SUPERVISOR makes the best recovery it can if it determines the time or date has changed. If the date has been changed, SUPERVISOR will act as if it were restarted after a period of time absent from the mix. It will load the SCHEDULE corresponding to the new "current day" and enter backtrack mode until it has performed or omitted all activities up to the current time of day. This behaviour can be controlled by the setting of the USE MODE ... FOR BACKTRACK. See [SUPERVISOR Control Commands](#) for details of the USE command.

If changing time or date is being performed on a regular basis for, say, contingency testing, it is strongly recommended that SUPERVISOR be QUIT after each time or date change. Further, the loading of "old" schedules means that SUPERVISOR could expend a great deal of time and effort backtracking any missed activities. This behaviour can be inhibited by disabling the automatic backtrack:

USE MODE NONE FOR BACKTRACK

Otherwise, if the time is changed forward (with the date remaining unchanged), SUPERVISOR will either enter backtrack mode, or simply perform the activities scheduled for the "skipped interval", depending on the amount the time was advanced. Any period over 3 minutes causes SUPERVISOR to enter backtrack mode.

If the system time is set backwards, SUPERVISOR will not repeat scheduled activities already performed.

Menu Based Scheduling

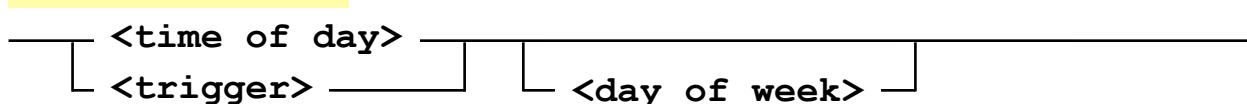
A suite of Opal programs which allow more advanced scheduling than is available using the default supervisor commands has been developed.

The interface is via a COMS window. Please contact Metalogic for more info on this.

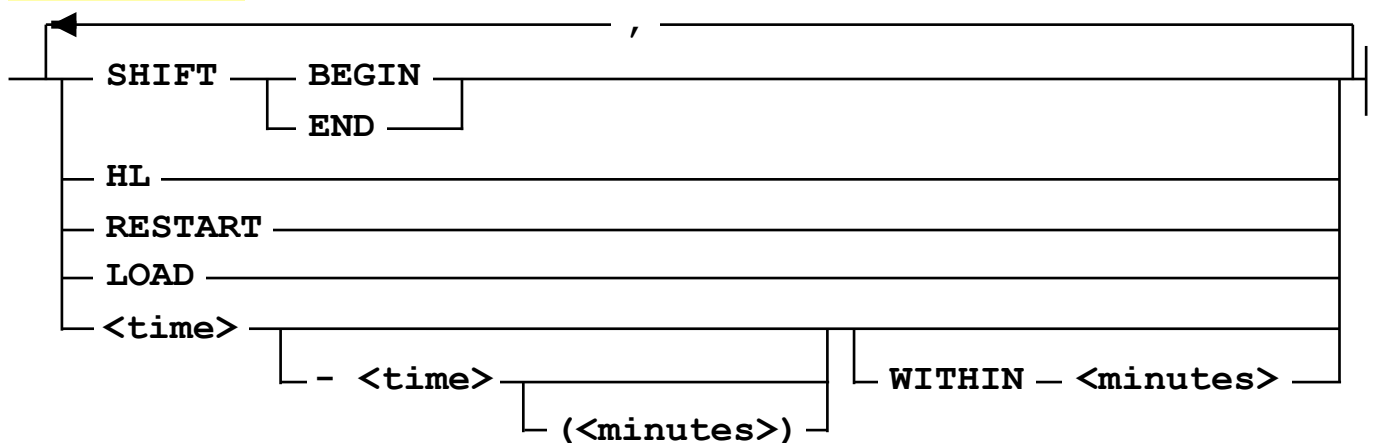
AFTER Command

The AFTER command accesses a simple, but powerful time scheduler. The various components which will be used in the subsequent description of the variants of the After command are described here.

<time specification>



<time of day>



The syntax item <time> is specified in military time format, using a 24-hour clock. Thus, 9AM is 900, 11.35PM is 2335, and so on. Midnight is 0000 or just 0; one minute to midnight is 2359. Lists of times are permitted, as are periodic intervals. The syntax item <minutes> is an integer specifying a number of minutes. After a time interval, it signifies the number of minutes between each repetition of the activity is repeated.

In a WITHIN clause, <minutes> gives the maximum number of minutes after <time> that the activity will be started, if there is a delay. If no WITHIN clause exists, there is no time limit on the activity.

Examples

AF 2315

means one time only at 11:15PM

AF 0900,1030,1200,1330,1500

AF 900-1500 (90)

means every hour and a half from 9AM until 3PM

AF 10-700 (10)

means every ten minutes from ten minutes past midnight to 7AM.

It is also possible to specify that some activity should be performed following a restart of Supervisor using RESTART. To select only the restarts caused by a Halt Load use HL. When an operator starts or finishes a shift use SHIFT.

For example:

```
AF RESTART
AF SHIFT BEGIN, SHIFT END
AF HL,1900
```

The special time LOAD indicates that the activity should be performed after a new days schedule has been loaded. This is the first activity to be performed for that day. Its normal use is to prevent a scheduled item being performed on a specific date.

For Example.

```
AF 1000 DAILY START DAILY/JOB
If we don't want this to run on December 12 2011 we could add
AF LOAD ON DATE 12/12/11 TT AF - 1000 START DAILY/JOB
```

By default, where <minutes> is not specified, SUPERVISOR will assume an interval of 1 minute. If a range of times is entered with no interval specified, SUPERVISOR will reject the command if the range spans more than 15 minutes except for interrogation with AF ?. To perform some activity each minute for more than 15 consecutive minutes, (<minutes>) must appear.

<trigger>

Trigger is followed by an identifier with an optional usercode.

Ex.

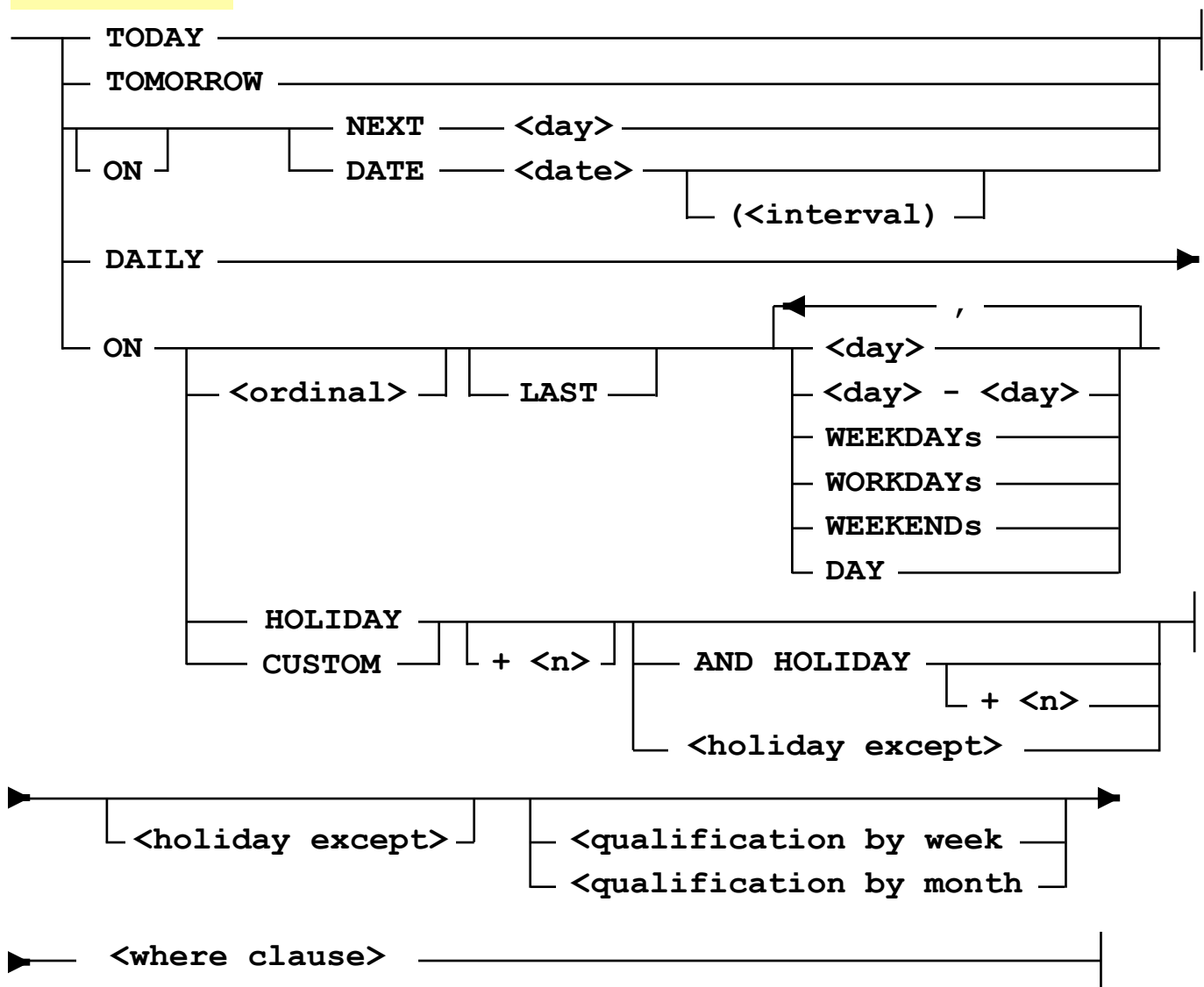
```
FOR LIVE/PW AF TRIGGER (BOB)TEST DAILY:START BOB/TEST
```

Once this is in the schedule, entering

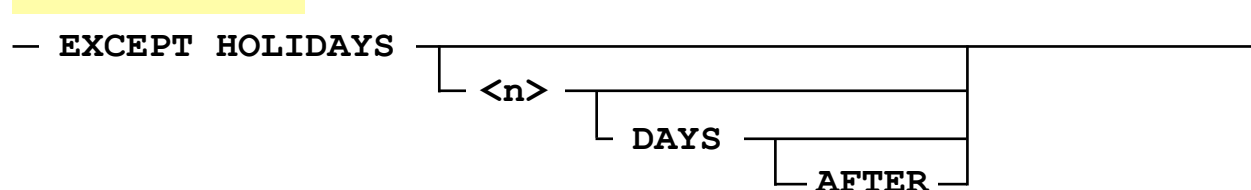
```
TT TRIGGER (BOB)TEST
```

will cause BOB/TEST to be started under the usercode LIVE. The intent of this change is to allow jobs to be stored in the schedule with appropriate Usercode and accesscode and then be 'Triggered' by a subsequent event, perhaps the successful completion of another Job. It also allows an operator to start a job under a specific usercode without needing access to the password for that user.

<day of week>



<holiday except>



The <time of day> of an activity can be further specified by day. The simplest cases are TODAY, TOMORROW, and NEXT <day>, where <day> is the name of a day in English:

MONDAY, TUESDAY, WEDNESDAY, THURSDAY, FRIDAY, SATURDAY, or SUNDAY

The format of the syntax item <date> is governed by the setting of the SUPERVISOR Option USDATES. If the option is set, the format of <date> is MM/DD/YYYY. If the option is reset, the format of <date> is DD/MM/YYYY.

The syntax item <interval> is an integer specifying (in days) how often the activity should be repeated.

If the DATE syntax is used, the requested activity is performed at one or more times on the specified date. If <interval> appears, it will be repeated at each <interval> days thereafter. If <interval> does not appear, it will be performed only on the specified date. <interval> must lie in the range 1-63, and unless it is divisible by 7 (<interval> MOD 7=0), the activity will appear in the schedule for each day of the week.

The other modifiers are ways of giving days of the week. Unless qualified further, they will be done on the specified days until further notice.

DAILY or DAY	selects all days of the week.
WEEKDAYS	means MONDAY - FRIDAY.
WORKDAYS	means WEEKDAYS EXCEPT HOLIDAYS.
WEEKENDS	means SATURDAY, SUNDAY.

Other days are specified by their English name.

Intermixed lists of <day>s and ranges of <day>s are permitted.

Examples

```
AF 2315 TOMORROW
AF 1030 NEXT TUESDAY
AF 2355-2359(1) ON DATE 31/12/1999
AF 2323 ON MONDAY, WEEKENDS
AF 0900-1000(15) ON TUESDAY-THURSDAY
```

<ordinal> is FIRST, SECOND, and THIRD or numeric equivalents like 1ST, 2ND, 3RD, and beyond, such as, 4TH, 11TH, 31ST etc. <ordinal> or LAST are used to specify a monthly cycle. Thus 7TH DAY refers to the 7th of each month, LAST DAY is the last day of the month, e.g. Dec 31st or Feb 28th. 2ND LAST DAY is Dec 30th or Feb 27th. Because they are based on monthly cycles, <ordinal> and LAST may not be used with the IN WEEK <week of year> construct.

By using the [HOLIDAY](#) command, it is possible to mark dates as holidays.

If the EXCEPT HOLIDAYS clause is used, the activity will be omitted on any date marked as a holiday. WORKDAYS is exactly equivalent to WEEKDAYS EXCEPT HOLIDAYS unless specific weekend days have been marked as work days.

The EXCEPT HOLIDAYS clause may be further modified by an integer

Example

```
AF 0100 ON MONDAY EXCEPT HOLIDAY 1 DAY AFTER:TT DO TEST
AF 0100 ON MONDAY EXCEPT HOLIDAYS 1 DAY:TT DO TEST
AF 0100 ON MONDAY EXCEPT HOLIDAY 1
```

Optionally, the 'EXCEPT HOLIDAYS n' clause will accept 'DAYS' or 'DAYS AFTER' following the number, so all of the above examples are equivalent. They would schedule an activity to run on every Monday unless that day was a holiday in which case it would run on the Tuesday.

By using the WORKDAYS command, it is possible to mark specified Saturdays and Sundays as work days.

The ON HOLIDAYS clause specifies that the activity will run only on Holidays.

ON HOLIDAY can be further modified by + <n> where <n> is an integer between 1 and 15. This indicates the activity is to run n days after a holiday.

The ON CUSTOM clause specifies that the activity will run only on dates marked as Custom dates using the [CUSTOM](#) command.

ON HOLIDAY can be further modified by + <n> where >n> is an integer between 1 and 15. This indicates that the activity will run n days after a Custom date.

An AND HOLIDAY clause may follow an ON CUSTOM or ON HOLIDAY clause.

Example

```
AFTER + 0001 ON CUSTOM+2 AND HOLIDAY+1:TT DO CUST_HOL
```

For the above, this activity will be executed on days when yesterday was a holiday and two days ago was a Custom date.

Except Holidays may also be used with ON CUSTOM and ON HOLIDAY.

Example.

```
AFTER + 0001 ON CUSTOM+1 EXCEPT HOLIDAY 2:TT DO CUST_HOL
```

If the CUSTOM+1 date is not a HOLIDAY, then the activity will be executed; if, however, CUSTOM+1 maps to a SUPERVISOR holiday then the activity is postponed 2 days into the future. So, where CUSTOM+1 is a holiday, the activity is effectively scheduled as CUSTOM+3.

```
AFTER + 0030 ON HOLIDAY+1 EXCEPT HOLIDAYS:TT DO HOL2
```

```
AFTER + 0030 ON HOLIDAY+1 EXCEPT HOLIDAYS 2:TT DO HOL2
```

In the first case, the activity will be actioned one day after any holiday UNLESS that day is also a holiday in which case the activity will not be done. It should be noted that EXCEPT HOLIDAY n is only permitted where 'ON HOLIDAY+n' has a non-zero offset.

In case two, the activity will be performed one day after any holiday UNLESS that day is a holiday in which case the activity will be performed 2 days later i.e. effectively ON HOLIDAY+3.

If no <qualification> occurs, SUPERVISOR assumes that only TODAY is meant, and the scheduled activity is performed only on the current date.

If <qualification> is found, and the requested list of days includes today (that is, if today's name or date occurs in the list), the activity will also be scheduled for the current day.

A mistake often made is to use
AF HL
(After any halt load today)
when
AF HL DAILY
(After any halt load)
is intended.

Examples

ON 7TH DAY

would execute on the 7th of every month.

ON LAST TUESDAY, WEDNESDAY EXCEPT HOLIDAYS OF MARCH, JUNE, SEPTEMBER

would execute once per quarter, on the last non-holiday day of that month that is a Tuesday or Wednesday.

ON 3RD LAST WORKDAY

ON 2ND WORKDAY IN SECOND WEEK

ON SECOND WORKDAY IN WEEKS 2

Usually the second Tuesday of every month, but if Monday or Tuesday is a holiday moves to Wednesday, or Thursday if both Monday and Tuesday are holidays, ...

ON SECOND WORKDAY IN WEEK 2

Not allowed – '2' refers to a <yearweeknumber>

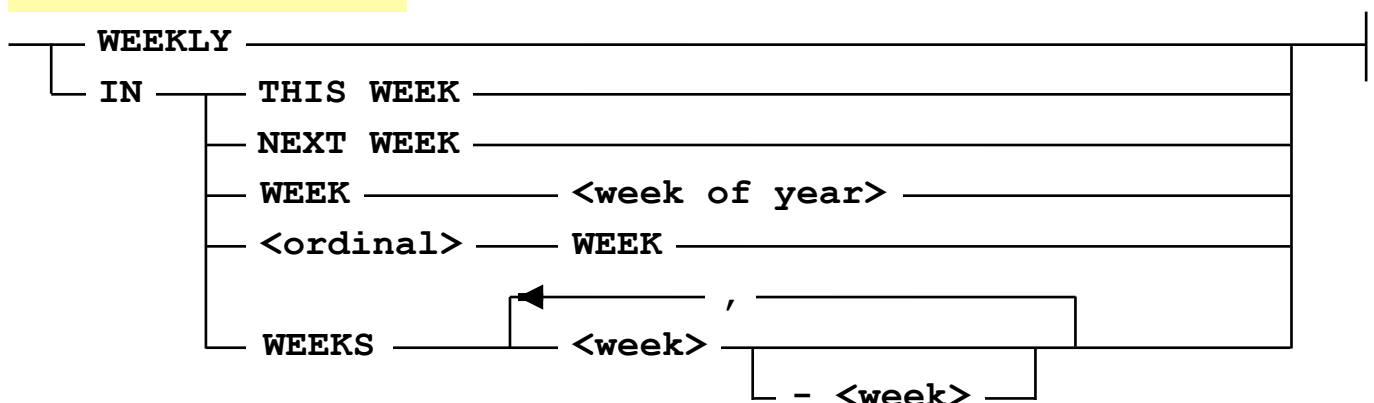
ON FIRST WEEKDAYS, SATURDAY

will pick the "most" qualified day specified. That is the first (or last) day of the month that fits the rest of the criteria, so it means the first non-Sunday of the specified month(s).

SECOND LAST WEDNESDAY - FRIDAY

When the month ends on a Wednesday, would refer to the last Friday of the month (the 2nd last qualified day, the second last day in the month that is a Wednesday, Thursday, or Friday).

<qualification by week>



The use of WEEKLY filters activity for each week of the month, is defined as Sunday->Saturday and is independent of month start. The WEEKLY modifier can *ONLY* be used after a WORKDAYS clause; all other usage will give an INVALID OPTION error.

For example:

```
AFTER +1100 ON 3RD WORKDAY WEEKLY:TT DO WORKODTS
```

For any month, the above AFTER would be executed on the third workday of each 7-day week, spanning Sunday->Saturday. For June 2002, the AFTERS would be executed for Wednesday 1st, 8th, 15th June.

Note that these date calculations will take account of any specified workdays or holidays. Therefore, a Sunday marked as a workday with the WORKDAYS command will become the first working day for an activity that uses the WEEKLY clause.

The syntax item <week of year> is an integer in the range 1-53. This construct is incompatible with use of <ordinal> and LAST in the day specification.

The syntax item <week> is an integer in the range 1-5 and refers to the week of the month.

An activity may also be scheduled on a particular week of the month or year. Such qualification uses the IN modifier, in combination with an <ordinal>, for example:

```
AF RESTART ON WORKDAYS IN FIRST WEEK  
AF 0900 DAILY IN 4TH WEEK
```

Lists of weeks by number are also permitted.

For example:

```
AF 0900 ON TUESDAY,WEDNESDAY IN WEEKS 1,3-4
```

The integer <week> must lie in the range 1-5. The WEEKS start on the first day of the month, and each subsequent week starts seven days after that. Thus to get the first Friday in December 1994, which is the second day of the month, use FRIDAY IN WEEKS 1. To get the last Friday of that month, use FRIDAY IN WEEKS 5. (These could be more simply referenced by FIRST FRIDAY and LAST FRIDAY respectively.)

December 1994							
S	M	T	W	T	F	S	
				1	2	3	WEEK 1
4	5	6	7	8	9	10	
11	12	13	14	15	16	17	
18	19	20	21	22	23	24	
25	26	27	28	29	30	31	WEEK 5

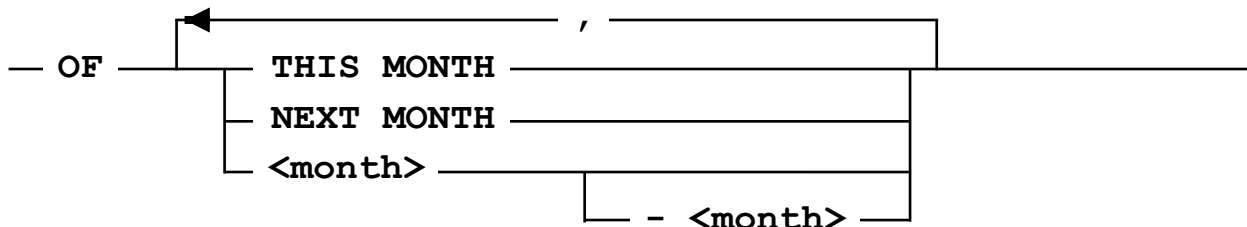
If no further qualification occurs, all months of the year are assumed. If a specific WEEK of the year is selected by its number, it follows the continental European convention of beginning the week on a Sunday. Thus if the first day of the year is a Thursday, there is no WEDNESDAY IN WEEK 1. The <week of year> for the WEEK clause must lie in the range 1 to 53.

For example:

AF 0900 ON MONDAY IN WEEK 26

No further qualification is permitted if the WEEK modifier is used.

<qualification by month>



Scheduled activities may be restricted to specific months of the year by means of an OF qualification. Months are referenced by their English names. The syntax item <month> is thus:

JANUARY, FEBRUARY, MARCH, APRIL, MAY, JUNE, JULY, AUGUST,
SEPTEMBER, OCTOBER, NOVEMBER, or DECEMBER

Lists and ranges are permitted.

For example:

AF HL DAILY OF NEXT MONTH
AF HL DAILY OF FEBRUARY, JUNE
AF HL DAILY OF APRIL-JULY, SEPTEMBER

Where no IN qualification occurs, all weeks of the month are assumed.

<where clause>

The <where clause> allows user defined restrictions to be added. The word WHERE is followed by the name of a system Situation. If all other conditions for the after are true then the Situation is evaluated. If it returns true the AFTER will run.

Ex. AF 1000 ON WORKDAYS WHERE Hol_Yesterday START MY/JOB

If Hol_Yesterday is defined as:

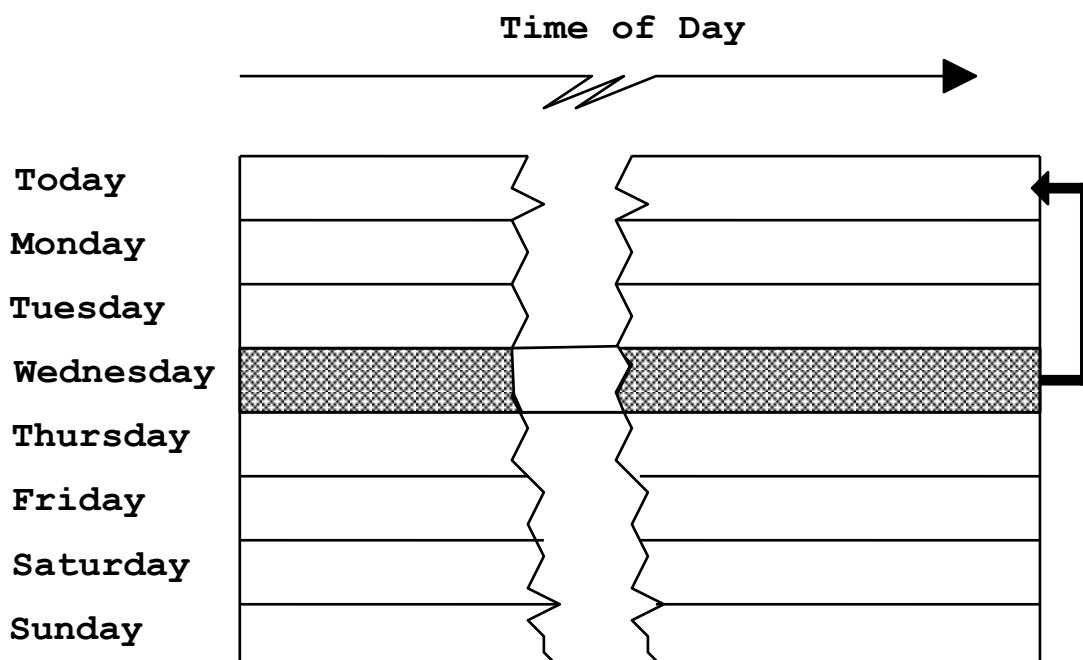
```
TT DEFINE + SITUATION Hol_YESTERDAY:  
Holiday(NewdateTStoday,-1)
```

Then the above AF would only run on Workdays where yesterday was a holiday.

A Description of the Schedule

All scheduled activities together form the current schedule. The schedule may be thought of as an 8 by 1444 matrix, where each element or slot contains the information required by SUPERVISOR to perform one or more scheduled activities. 1440 is the number of minutes in one day, plus four slots for: HALTLOAD, RESTART, SHIFT BEGIN and SHIFT END. The eight rows represent each day of the week plus one for TODAY.

At midnight SUPERVISOR copies all slots from the new day of the week to the row for TODAY. All prior information in the TODAY row is discarded. AFTER (AF) commands that default to TODAY, alter only this row; but those specifying a particular day or days may also update entries in TODAY's row.



After midnight on Monday, the row for Tuesday is copied into the row for TODAY.

AFTER <time> TODAY refers to one date only.

AFTER <time> ON TUESDAY refers to each and every Tuesday.

Scheduling an activity

— **AF**ter + **<time specification>** : **<text>** —

The **<text>** is normally something which could be entered at an ODT. SUPERVISOR checks the syntax of the command when the AF is specified, and if it detects that it is invalid, the AF is rejected. The **<time specification>** gives one or more times when the **<text>** should be given to the system. The effect of scheduling an activity is the same as if, at the time or times specified by the **<time specification>**, the **<text>** had been entered directly from the same origin. If **<text>** is a SUPERVISOR command, then it will be acted upon by SUPERVISOR, if not, it will be passed to the system (to the MCP).

For example, to close queue 10 at 11pm today:

```
TT AF 2300 MQ 10 ML 0
```

If the "+" form of the After command is used then the activity will not be scheduled for a time where it is already present.

The text following the **<time specification>** can be any valid system command, WFL, or SUPERVISOR command (including another AFTER command). Most SUPERVISOR commands can be input by using a TT prefix as if it came from the ODT. Some commands can be used without a TT prefix, and this allows SUPERVISOR to encode the command in a more concise way. ??RUN commands may not have a TT prefix. Some commands are not allowed in the text, with or without a TT prefix. These commands are flagged with a NO in the 'With AF ?' column of the command tables.

See the following command tables: [Basics](#), [Control](#), [System](#), [Opal](#)

Certain scheduled activities will only be done once for any one-minute, even if scheduled several times in that minute, as long as they have no WITHIN, IN, OF, or ON part.

These commands are:

```
SAVE SCHEDULE
SAVE USERDATA
TL
QUIT SUPERVISOR
PRINT SCHEDULE
```

It is also worthy of note that any of these commands may be preceded by TT (e.g. the second TT in "TT AF **<time and date>** TT SAVE SCHEDULE") to avoid this constraint. Said another way, DO NOT use the TT in the scheduled activity if you want to stay away from redundant executions of these commands. The effect is multiplied during backtrack.

If you leave the TT out of the scheduled activity, only one of each such activity will be done during backtrack. Also, the non-TT form of the above commands will ignore

The appearance of lookup functions (see OPAL Manual for full detail on Lookup Functions) within <text> may cause SUPERVISOR to substitute, just prior to passing to the system, e.g. a list of mix numbers. When a scheduled activity is about to be performed, and it is not a SUPERVISOR command, the text is re-examined to evaluate any lookup functions that may be present.

AF 2100 #[MX=SYSTEM/CANDE] SM QUIT

SUPERVISOR eliminates trailing and leading blanks from any text associated with a scheduled activity, before storing it in the schedule. This eases deletion at a later stage.

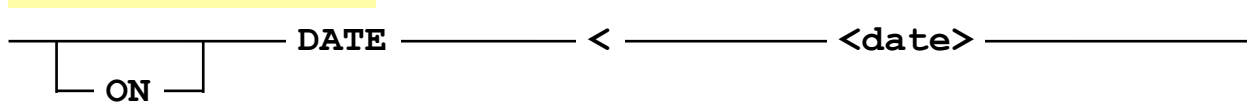
Examples

AF 0800: MQ 10 MIXL 12
AF 1200-1330(10) ON TUESDAY #[MX=SYSTEM/CANDE] SM SS ALL PM AT 1:30
AF + 1200 ON TUESDAY IN FIRST WEEK: PO #[PK=CANDEUSERS]
FOR U/P ACCESS AC/AP CHARGECODE OPS AF SHIFT BEGIN DAILY:STARTJOB X
AFTER + 0831 WITHIN 30: AT BLUE MIXL 0
AFTER HL ON WEEKDAYS: NET -
AF 0800,1200,1600,2000 ON WEEKDAYS,SATURDAYS EXCEPT HOLIDAYS: TL 1
AF 0 DAILY: TT AF + HL: NET +

```

graph LR
    A[After] --- B["- - - <time specification>"]
    B --- C["<text>"]
    B --- D["<obsolete date spec>"]
    C --- E["@"]
    E --- F["<integer>"]
    E --- G["="]
    G --- H["<integer>"]
    D --- I["@="]
    I --- J["<integer>"]
    I --- K["<integer>"]
  
```


<obsolete date spec>



Will find any entries in the schedule which were set for a specific date (using the ON DATE <date> construct) and where that date is less than the date provided.

If <text> is present, it must exactly match the <text> of the activity to be deleted. For example:

```
AF - 2300 OP + NODUMP
```

Where the "@ <integer>" is used, the activity which is to be deleted is found by looking at the earliest time in the <time specification> and substituting the <text> for the activity of that number for that time. The delete then continues as if the <text> had been entered. The number of the activity at a particular time is given by the AFTER ? response.

Example

```
AFTER - 2300 @ 2
```

Where the "@=" is used with a normal <time specification>, all entries at the specified time are deleted. This variant must be prefixed by a FOR modifier using a privileged usercode. An additional restriction is that the <time specification> may not contain a range of times and <qualification> must be absent, TODAY or ON <day> only because it must reference one unique schedule slot

Example

```
FOR PRIV/USER AFTER - 1800 ON FRIDAY @ =
```

Care must be taken when using deletion by numbers, especially when <time specification> specifies a range of times. SUPERVISOR treats TODAY as the earliest day. To ensure the number is correct, put the <time specification> in an AF ? and read the number from the first time in the response. If no activity with the requested number is found at that time, it returns a message like:

```
THERE IS NO ACTIVITY NUMBER 3 AT 2340 ON FRIDAY
```

An <obsolete date spec> is used to delete all entries, which have passed their execution date.

Examples

```
AF - 1700 ON TUESDAYS @ 2
```

```
AF - 0900 ON WORKDAYS OF AUGUST FOR PRIV/USER: ??RUN RT/CONTROL
```

```
AF - 0-2359 ON DATE < 25/08/2008 @ =
```

Simulating an add or delete

To assist the Metalogic web browser interface for schedule maintenance, the AFTER command has been modified to allow requests to be verified without updating the

Time based example

```
af ? 0100-0300 on friday
----- PENDING SCHEDULE after 01:00 ON FRIDAY -----
1: TT DO STATS_CHANGE
2: OP -27

----- PENDING SCHEDULE after 01:01 ON FRIDAY -----
1: ON 1ST DAY TT DO FRED

----- PENDING SCHEDULE after 01:59 ON FRIDAY -----
1: TT DO SMTP_HEARTBEAT

----- PENDING SCHEDULE after 02:59 ON FRIDAY -----
1: TT DO SMTP_HEARTBEAT

----- PENDING SCHEDULE after 03:00 ON FRIDAY -----
1: TT DO META_REMOVE
2: TT DO META_COPYCAT
```

HL Example

```
AF ? HL DAILY
----- PENDING SCHEDULE after HALTLOAD ON SUNDAY -----
1: MQ 0 MIXL= 50
2: MQ 255 ML 1
3: MQ 50 ML 1
4: MQ 2 ML 50
5: ??RUN METALOGIC/SUPERVISOR/WAITWATCHER
6: ??RUN SYSTEM/COMS
7: TT DO NET_ICMP

----- PENDING SCHEDULE after HALTLOAD ON MONDAY -----
1: MQ 0 MIXL= 50
2: MQ 255 ML 1
3: MQ 50 ML 1
4: MQ 2 ML 50
5: ??RUN METALOGIC/SUPERVISOR/WAITWATCHER
6: ??RUN SYSTEM/COMS
7: TT DO NET_ICMP

----- PENDING SCHEDULE after HALTLOAD ON TUESDAY -----
1: MQ 0 MIXL= 50
2: MQ 255 ML 1
3: MQ 50 ML 1
```

Example

```
AF ? date < 23/4/2008
----- PENDING SCHEDULE after 07:00 ON MONDAY < 23/04/2008 -----
1: ON DATE 07/04/2008 TT UO - 46

----- PENDING SCHEDULE after 12:31 ON MONDAY < 23/04/2008 -----
1: ON DATE 19/11/2001 TT DO X

----- PENDING SCHEDULE after 12:00 ON TUESDAY < 23/04/2008 -----
1: ON DATE 11/01/2005 TT DO X1

----- PENDING SCHEDULE after 07:00 ON WEDNESDAY < 23/04/2008 -----
1: ON DATE 02/04/2008 TT UO + 46

----- PENDING SCHEDULE after 19:00 ON WEDNESDAY < 23/04/2008 -----
1: ON DATE 09/04/2008 TT DO FRED

----- PENDING SCHEDULE after 10:00 ON THURSDAY < 23/04/2008 -----
1: ON DATE 05/10/2006 TT DO Z

----- PENDING SCHEDULE after 12:00 ON THURSDAY < 23/04/2008 -----
1: ON DATE 01/01/2004 TT DO XX
```

No time , with wildcard example

```
AF ? =SAVE=
----- SCHEDULE after 04:00 TODAY @ 04:00:00 24/04/2008 -----
3: TT DO SAVE_SCHEDULE_AS : (OK)
----- PENDING SCHEDULE after 20:15 TODAY -----
1: SAVE SCHEDULE
```

Associating Scheduled Activities with Usercodes

Scheduled activities may be "locked", that is, made not subject to cancellation without knowledge of a valid password corresponding to the locking usercode. This may be done by means of the FOR Modifier. The reserved word "FOR" followed by a valid usercode and password combination may precede or follow the "AF <time specification>" string. At the time of entry of the command, SUPERVISOR will verify the usercode and password combination and store the usercode (but not the password) along with the text in the SCHEDULE file. When the command is to be executed, SUPERVISOR verifies that the usercode is still valid (still DEFINED and not suspended) by means of the USERDATA intrinsic.

Any attempt to delete the activity from the schedule will be rejected unless the same usercode and a valid password (but not necessarily the same one) are also supplied by means of a FOR Modifier. The sole exception to that rule is that a privileged user can, by specifying a privileged usercode and some valid password, delete any scheduled activity "locked" with a non-privileged usercode. In this case, two FOR qualifications must appear, the first specifying the usercode of the privileged user and the second specifying only the usercode by which the activity was locked.

For example, user A, a non-privileged user, might schedule a job for some time in the future

```
AF 2200 FOR A/B STARTJOB MYJOB ON TESTPACK
```

which could subsequently be cancelled with

```
AF - 2200 FOR A/C STARTJOB MYJOB ON TESTPACK
```

even though the password has changed from B to C since the time the activity was entered.

The privileged user PRIV, who wants to delete this activity would use

```
FOR PRIV/USER FOR A AF - 2200 STARTJOB MYJOB ON TESTPACK
```

In this case, PRIV did not know the password of A, but was still able to delete the activity. However, a privileged user may not delete an entry for another privileged user. Such an entry can only be deleted by the originating usercode. Any scheduled activity, with an associated usercode, causes SUPERVISOR to verify the usercode is still valid at the time the activity is to be performed.

If it is valid and the command is not a SUPERVISOR command, the usercode will be associated with the string sent to CONTROLLER in the same way as CANDE

commands are restricted to the user's own usercode. This mechanism resembles also the action of the TERM USER system command. Thus jobs started as a scheduled activity will be started under the locking usercode and system commands can only affect programs running under the locking usercode. The FOR modifier may also be accompanied by the CHARGE or ACCESS modifiers, to associate chargecodes and accesscodes to the execution. These are also validated both on schedule entry and execution.

FOR PRIV/USER CHARGE GRP/PROG/APP ACCESS APP/APW STARTJOB X

Any MCS, such as SYSTEM/RJE, has the possibility of entering system commands from terminals, also has the possibility of communicating with SUPERVISOR and scheduling activities. Messages from these terminals with an associated usercode (the log-on usercode to RJE) will be treated by SUPERVISOR as if an implied FOR Modifier preceded the actual text of the command unless they have an explicit FOR modifier.

Scheduled activities with no associated usercode may not be deleted from the schedule by a source with an associated implicit or explicit usercode (TERM USER on the system console, RJE log-on usercode or an explicit FOR qualification).

Timestamps and Activity Postscripts

Each time slot in the SCHEDULE with an activity has some extra information stored within it. There are three timestamps (time and date):

- CREATIONTIME: when an entry was first made for this time.
- AMENDTIME: when an entry was last altered.
- ACCESTIME: When this time slot was last accessed.

These timestamps are printed for each slot by the report generated by a "PRINT SCHEDULE" command (unless the BRIEF option is used). Every slot in the schedule also has an additional field, the POSTSCRIPT field, which contains flags that are set if exception events prevented the successful performance of the activity the last time its execution was attempted. Activities that have been successfully executed are marked as "OK". Scheduled activities may not be performed for a variety of reasons, which are as follows:

- QUEUED: an activity which runs a utility (e.g. SYSDIR) was delayed because another utility was active. This postscript is cleared when the utility finally starts.
- FAMILY UNAVAILABLE: an activity needed a pack family which was not on-line.
- FILE UNAVAILABLE: a file needed by an activity was not available.

- **INVALID ACCESS:** the accesscode of the scheduled activity was no longer valid at the time SUPERVISOR attempted to perform it. This can arise if the accesscode was deleted.
- **INVALID CHARGE:** the chargecode of the scheduled activity was no longer valid at the time SUPERVISOR attempted to perform it. This can arise if the chargecode was deleted.
- **TIMED OUT:** a REMIND command was not acknowledged by the operator within a 5 minute period.
- **SKIPPED:** an operator responded "TT SKIP" to the prompt during a "BACKTRACK" operation. This occurs when SUPERVISOR starts after not running when an activity should have been run.
- **INVALID USER:** the usercode of the scheduled activity was no longer valid at the time SUPERVISOR attempted to perform it. This can arise if the usercode was deleted or suspended.
- **JOB OK,#<job number>:** A WFL job was started, <job number> is the assigned jobnumber of the job started.
- **NOT TODAY:** the scheduled activity was for a specific day or date that has now passed. In these cases, SUPERVISOR errs on the side of caution by not running the activity.
- **NONEXECUTABLE OR INCOMPATIBLE:** SUPERVISOR cannot run the activity because of some inconsistency within the designated OPAL program.
- **UNUSABLE FILEKIND:** a file has been specified which is of a filekind not recognised by SUPERVISOR.
- **SYNTAX ERROR IN WFL :** The activity was a WFL job, which was found to have a WFL syntax error on initiation.
- **WFL TO BE STARTED IS NOT AVAILABLE :** The activity was the START of a WFL job, but the file to be started was not found at the time of execution. As this is checked when the entry is scheduled, it must have been removed afterwards.

DO commands executed by an AFTER will generally only be marked with a 'postscript' of 'OK' or 'COMMANDERROR' as seen in an AF ? response. In the event that a scheduled DO command invokes a WFL job via the ODT statement then the postscript of the activity will be updated with the job number and status of the WFL job (as seen with a WFL activity explicitly scheduled by an AFTER) at run time.

Note that this update only occurs for the first WFL job that is processed by the DO since it is not feasible to hold information for more than one WFL entity in an AFTER record.

The POSTSCRIPT (if any) is reported in response to an AF ? command for any time that has passed and is also printed in the report generated by a PRINT SCHEDULE

command.

Typical values include JOB OK, JOB SYNTAX, BAD USERCODE, INVALID ACCESS etc. There is also an additional value, COMMANDERROR, which identifies if an AF TT command has failed e.g. a TT DO might fail because the ODTSEQUENCE might be absent.

```
AF ? 1930-1934
----- SCHEDULE after 19:30 TODAY @ 19:30:54 28/02/2008 -----
1: STARTJOB (IPP)TEST/JOB ON DEV : (JOB OK,#54131)
----- SCHEDULE after 19:31 TODAY @ 19:31:48 28/02/2008 -----
1: FOR (TEST) STARTJOB (IPP)TEST/JOB ON DEV : (INVALID USERCODE)
----- SCHEDULE after 19:32 TODAY @ 19:32:36 28/02/2008 -----
1: FOR (TEST1) ACCESS IPP STARTJOB (IPP)TEST/JOB : (INVALID ACCESS)
----- SCHEDULE after 19:33 TODAY @ 19:33:12 28/02/2008 -----
1: FOR (TEST2) CHARGE IPP STARTJOB (IPP)TEST/JOB : (INVALID CHARGE)
----- SCHEDULE after 19:34 TODAY @ 19:34:06 28/02/2008 -----
1: TT DO ZZ : (OK)
```

When SUPERVISOR logging is active, the LOG command will show all AFTER commands being processed at the activity time and report each success or failure.

The AFTER context has been extended to provide these AFTER results when used with the EVAL command. The POSTSCRIPT and JOBNUMBER attributes provide for WFL jobs whether it started successfully and, if so, the associated job number. Note that JOBNUMBER is also available if the WFL activity fails with syntax errors.

The AFTER response for the daily schedule will now show WFL job numbers where appropriate for both AF ? and PRINT SCHEDULE commands.

This facility could, for example, allow SUPERVISOR Opal programs to report AFTER failures on a daily basis and email any exceptions.

```
EV(AFTER:VALID(POSTSCRIPT)) DO (SHOW(ADDTEXT 50,,POSTSCRIPT 12,,JOBNO))

FOR TEST2 CHARGE IPP AFTER + 1933:STARTJOB (IPP)TE INVCHARGE      0
FOR TEST1 ACCESS IPP AFTER + 1932:STARTJOB (IPP)TE INVACCESS      0
FOR TEST AFTER + 1931:STARTJOB (IPP)TEST/JOB ON DE BADUSERCODE    0
AFTER + 1930:STARTJOB (IPP)TEST/JOB ON DEV          JOBSYNTAX      60356
FOR IPP AFTER + 1929:STARTJOB (IPP)MK/JOBDEL ON DE  JOBOK          60348
AFTER + 1929:PRINT SCHEDULE                          JOBOK          60348
AFTER + 1929:SAVE SCHEDULE                          JOBOK          60348
FOR TEST2 CHARGE IPP AFTER + 1928:STARTJOB (IPP)TE  JOBOK          60341
FOR TEST1 ACCESS IPP AFTER + 1927:STARTJOB (IPP)TE  JOBOK          60339
FOR TEST AFTER + 1926:STARTJOB (IPP)TEST/JOB ON DE  JOBOK          60335
AFTER + 1925:STARTJOB (IPP)TEST/JOB ON DEV          JOBOK          60333
FOR IPP AFTER + 1924:STARTJOB (IPP)MK/JOBADD ON DE  JOBOK          60327
AFTER + 1923:TT DO NOSUCHACTIVITY                   COMMANDERROR 0
AFTER + 1922:SAVE SCHEDULE                          AFDONE          0
AFTER + 1921:DISPLAY("SIMPLE WFL TEST")             AFDONE          0
INLINE_AFTER_14+INLINE_14 WILL BE EVALUATED
```

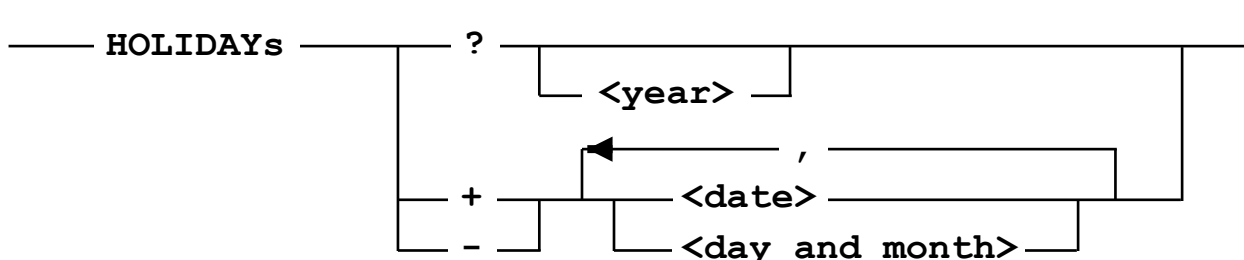
The above EVAL could be easily changed to allow failed jobs to be collected and emailed at 2359 each night. See HELP ATTR POSTSCRIPT for details of all the possible values for this attribute.

SUPERVISOR-invoked commands such as JAM, SYSDIR and SYSLOG will now return JOBOK postscripts if successfully run and the associated job number will be available. This also applies to any RUN commands because such AFTER activities are jacketed by a BEGIN JOB..END JOB. Note that ??RUN entries that are successfully executed are exempt and cannot show a job number.

An example (partial) of PR SCHED output is shown below:

```
*AFTER 1254      :TT DO OPL_SKIP : (Skipped)
                  :TT DO FRED : (Skipped)
                  [RECORD TIMESTAMPS. CREATED:1252 18/11/93,AMENDED:1252 18/11
*AFTER 1258      :DISPLAY("GOING") : (OK)
                  [RECORD TIMESTAMPS. CREATED:1252 18/11/93,AMENDED:1252 18/11
*AFTER 1415      SAVE SCHEDULE
*AFTER 1627      :TT DO OPL_SKIP : (Skipped)
                  [RECORD TIMESTAMPS. CREATED:1626 18/11/93,AMENDED:1626 18/11
*AFTER 1628      :TT DO OPL_SKIP : (Skipped)
                  [RECORD TIMESTAMPS. CREATED:1626 18/11/93,AMENDED:1626 18/11
*AFTER 1629      :TT DO OPL_SKIP : (Skipped)
                  [RECORD TIMESTAMPS. CREATED:1629 18/11/93,AMENDED:1629 18/11
*AFTER 1631      :TT DO OPL_SKIP : (Timed out)
                  [RECORD TIMESTAMPS. CREATED:1630 18/11/93,AMENDED:1630 18/11
AFTER 1700      :TT DO META_WARN
                  [RECORD TIMESTAMPS. CREATED:1721 21/04/93,AMENDED:1721 21/04
AFTER 1715      SAVE SCHEDULE
AFTER 1800      ON DATE 12/11/92 :TT DO FRED
                  [RECORD TIMESTAMPS. CREATED:1654 11/11/92,AMENDED:1654 11/11
```

HOLIDAY Command



The HOLIDAY command adds (HOLIDAY) or deletes (HOLIDAY -) a date from the list of holidays, or displays (HOLIDAY ?) the current list.

There are two types of holidays – those where the date of the holiday varies within each year (e.g. Easter) and those which always occur on a fixed day of a fixed month (e.g. Christmas). The HOLIDAY command accepts either type. Movable holidays must be assigned by giving the date for each year.

The format of <date> depends upon the SUPERVISOR Option USDATES. The normal DD/MM/YYYY is used if USDATES is reset, and the United States variant MM/DD/YYYY if USDATES is set. Fixed holidays may be assigned by entering the day and the month (in the form DD/MM or MM/DD depending upon USDATES).

The list of holidays is held in the SCHEDULE file and is thus backed up by a SAVE command, and restored with a RELOAD command.

Should today's date be entered as a holiday, any activities flagged with the EXCEPT HOLIDAYS flag will be omitted for the rest of the day. Otherwise, the test for a holiday is made at midnight.

Example interrogation

TT HOLIDAYS ?

----- HOLIDAYS -----

JANUARY	1
JANUARY	2
MARCH	25, 2005
APRIL	10, 2009
APRIL	13, 2009
MAY	4, 2009
MAY	25, 2005
DECEMBER	26

If the optional <year> is specified in an interrogation, only holidays in that year are reported and a total of the holidays for that year is shown.

This is intended to make it easier to confirm that all of the holidays for the year supplied have been specified.

Ex.

HOLIDAYS ? 2009

----- HOLIDAYS 2009 -----

JANUARY	1
JANUARY	2
APRIL	10, 2009
APRIL	13, 2009
MAY	4, 2009
DECEMBER	25
DECEMBER	26
Total Holidays	:7

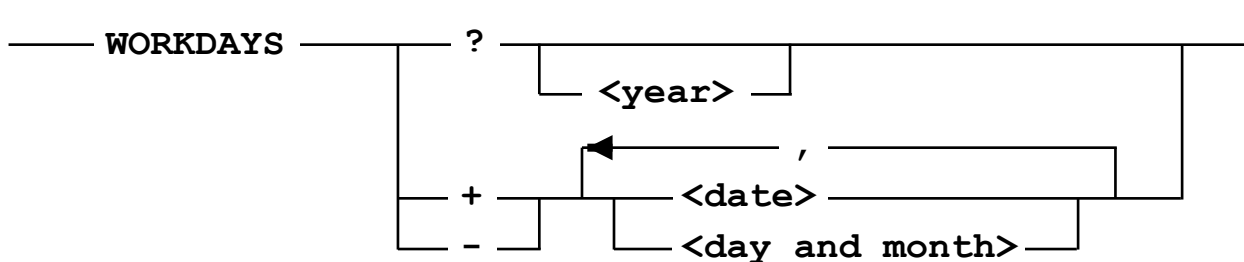
Examples setting

HOLIDAY 25/12,26/12

HOLIDAY - 09/04/93

HOLIDAYS 04/07,01/04/1999

WORKDAYS Command



All reference to activities using the clause 'ON WORKDAYS' will include any additional weekend working days that have been assigned by the WORKDAYS command. Only dates that map onto Saturday and Sunday can be assigned by the WORKDAYS command; other days will generate a syntax error.

Example

WORKDAYS + 1/8/2005

THAT DATE IS NOT A SATURDAY OR SUNDAY

Also, Supervisor will automatically add the current year if a date is given with only day and month values. For adding new WORKDAYs, the '+' is not required but is now allowed for both the HOLIDAYS and WORKDAYS commands.

A specific date cannot both be a WORKDAY and a HOLIDAY, giving the error:

WORKDAY CANNOT BE A HOLIDAY

The format of <date> depends upon the SUPERVISOR Option USDATES. The normal DD/MM/YYYY is used if USDATES is reset, and the United States variant MM/DD/YYYY if USDATES is set. Fixed workdays may be assigned by entering the day and the month (in the form DD/MM or MM/DD depending upon USDATES). If a year is not specified, the current year only is assumed.

The list of workdays is held in the SCHEDULE file and is thus backed up by a SAVE command, and restored with a RELOAD command.

Further, when using any 'ON WORKDAYS' clause in an AFTER, Supervisor will now make additional schedule entries for both Saturday and Sunday. This means that the response will show 7 or 8 activities scheduled instead of 5 or 6 (depending if the AFTER was valid for today).

A boolean OPAL attribute, WORKDAY, will return TRUE if today's date is a working day i.e. not a HOLIDAY, a normal week day or a Saturday or Sunday marked by the WORKDAYS command. The NEXTWORKDAY function will now include Saturday and Sunday dates assigned as working days.

Example interrogating

TT WORKDAYS ?

----- WORKDAYS -----

MARCH 5, 2005

MARCH 21, 2009

APRIL 12, 2009

JUNE 1, 2002

JUNE 2, 2002

If a year is specified in WORKDAYS ?, only workdays in that year are reported and a total of the workdays for that year is shown.

This is intended to make it easier to confirm that all of the workdays for the year supplied have been specified.

Ex.

```
WORKDAYS ? 09
----- WORKDAYS 2009 -----
MARCH          21, 2009
APRIL          12, 2009
Total Workdays :2
```

Examples setting

```
WORKDAYS + 21/4/02, 8/6, 17/8
WORKDAYS 13/7/2002
```

HOTLINE and RECORDER

Although SUPERVISOR makes the best use of standard A-Series resources for reporting (messages to system consoles or REMOTESPOs, log entries, printouts), there are times when it may be desirable to stream information to one or more user-specified destinations in a customised format. To this end, METALOGIC have implemented the HOTLINE and RECORDER mechanisms.

Within an OPAL program, it is possible, by use of the RECORD verb, to direct output to a specific disk file. In addition, the information can be directed to a BNA or TCPIP port file interface. By default, RECORDER and HOTLINE communications operate using BNA port files and multiple RECORDER clients can connect to one or more HOTLINEs. This means that a single HOTLINE program can receive alerts from multiple A-Series systems, each running SUPERVISOR and RECORDER.

However, this mode of operation can be switched to TCPIP by changing one configuration variable on each system. Using TCPIP port files allows, for example, critical system events to be detected and alert messages to be generated that are then directed through port files to a non A-Series environment. Some SUPERVISOR users have implemented automatic operator alert systems using this mechanism to redirect OPAL output to IBM PC/Windows-type environments; others have linked into paging systems so that voice messages can be generated and telephoned to off-site personnel.

METALOGIC provides sample, usable RECORDER and HOTLINE programs in source form on the software release media. The purpose of this section is to document the user interface to the HOTLINE program, and give guidelines for customising both programs.

Before customising Recorder or Hotline please contact Metalogic with your requirements. We will often add user requested features to these programs, avoiding future maintenance problems.

OPAL RECORD statement

```
— RECORD  [     <integer>     ] - ( <OPAL string> ) ————  
              <mnemonic>    
```

The RECORD statement allows textual information, alerts,, etc. to be passed by Supervisor or Flex where it can be written to disk or, to the outside world, via a port file.

The OPAL verb RECORD passes a string from an ODTSequence to a program called RECORDER, generally for the purpose of logging some condition to a disk file or to pass information to a HOTLINE or alert program.

Example

```
RECORD[6] ("This is a test message")
```

By default, RECORDER uses predefined file type mappings for each RECORD index:

```
0 map to header files (HDR)
0,1,2,3,4,5,11 map to disk files (DISK)
6,7,8,9,10,11 map to port files (PORT)
50,51,52,53map to variable length files (VAR)
```

However, it is also possible to configure or override the above settings with user-specified mappings. This is controlled by a Magus configuration variable called REC_FILEMAP that is maintained by the Supervisor REC NAMES command (please refer to the [Rec command](#) for more detailed information).

For example, defining the mnemonic TESTMNEM:

```
RECORD[TESTMNEM] ("This is a test message")
```

If the RECORD mnemonic was invalid, the OPAL compiler will give a syntax error:

```
Unknown RECORDER class mnemonic - TESTMNEM
```

The following might be a typical responses from a TT REC NAMES command:

```
TT REC NAMES

--- 0 of 18 Disk Files currently open ---
RECORD [01] mapped to CRITCL          DISK,PORT
RECORD [03] mapped to EOJ             DISK,PORT
RECORD [04] mapped to RSVP           PORT
RECORD [05] mapped to EOT             PORT
RECORD [06] mapped to CONFIG          DISK,PORT
RECORD [13] mapped to HEROIX          PORT=10.0.0.26

Default DISK files NOT mapped 0,2,11
Default PORT files NOT mapped 9,10,11,47
Default VAR files NOT mapped 50,51,52,53
```

In the response shown earlier, a RECORD[4] statement could now be coded in OPAL as RECORD[RSVP]; its output will be directed through a PORT file instead of a DISK file (which is the normal default). Similarly, RECORD[6] can be written as RECORD[CONFIG] and its output will go to DISK as well as the default PORT file.

Any changes to the name configuration, using the REC NAMES command, are applied immediately; RECORDER does not require a restart.

Recording to PORT files

RECORD sub files 6 thru 11, by default, are used to send messages to external TCPIP or BNA clients via the network. The Metalogic HOTLINE program is a customisable program that can receive RECORDER generated messages from multiple Clearpath systems using either protocol. The HOTLINE program is discussed in more detail later in this chapter.

RECORDER now supports a specific TCPIP port file to a nominated IP address.

The REC NAME syntax has been extended to allow the specification of an IP address.

Example:

```
TT REC NAME + HEROIX PORT=10.0.0.26
```

A TT REC QUIT should be done after setting or changing any PORT mappings.

Record messages passed to this port have no additional data added to (i.e. Not timestamp or Hostname). The port is closed after each write.

This change was made to enable Supervisor to create HEROIX ROBOEDA events. The layout of the events is controlled by Opal routines which are freely available to anyone interested in passing events to the ROBOEDA application.

The TT REC TCPIP command is used to specify the port number to be used (2919 is the default value for ROBOEDA).

Recording to disk files

In the default environment, RECORDER creates up to six disk files, 0 through 5, at one time. To RECORD to files 1 through 5, enter the file number in brackets and the text to be logged in parentheses.

For example:

```
Record [3] (TimeDate(YYYYMMDDHHMMSS) , , "User" , , User , ,  
"program" , , Name , , "security violation (COMPSEC) " ) ;
```

RECORDER does not automatically log either the time and date, or the identification of the originating OPAL. If these are required they should be included in the text (as shown in the example) so that you can identify the messages. This is especially important if several of your OPALs perform RECORDs to the same file number.

Another example where record index mapping has been established and RECORD[18] is mapped to DISK with an <id> of ALERT.

Might be:

```
Record [ALERT] ("Alert message text")
```

The file naming convention used when the disk files are created is detailed later in this section. See [RECORD file names](#).

Recording to disk file 0

Recording to disk file 0 can be accomplished in the same way as recording to disk file 1 to 5 by substituting a file index of 0. RECORDER treats file [0] in a special way and adds an additional line, showing the time of day and identifying the originating OPAL, to the output by RECORDER.

The BLOCKSIZE of the file is 450 words and the file is CLOSED after each record entry is written. Furthermore, if the log string begins with the text "QUIT", RECORDER will go to EOT after logging the entry.

Recording to disk file 11

Recording to file index number eleven (11) is a special case where RECORDER will both send the message text to HOTLINE, using a PORT file, and at the same time log it to a disk file.

The disk file name does not follow the usual convention described later in this section but instead has the general form:

```
*SUPERVISOR/RECORD/<dayname>/HOTLINE
```

This allows a site to keep a record of messages sent to HOTLINE.

Recording to disk files 50,51,52,53,54,55 (type VAR)

RECORDER will now create variable length record disk files when a RECORD with a file index in the range 50 to 55 is performed. Each disk file record will have length equal to the length of the string recorded. These files are intended for transfer to PCs for later processing. The only way to view them on the A-series system is by using the SYSTEM/DUMPALL utility.

Messages written to files of type VAR have a maximum length of 9133 characters; messages longer than this will be truncated.

Causing RECORDER to quit

As well as using the TT REC QUIT command, issuing a RECORD[0] with a message starting with the string "QUIT" will cause RECORDER to terminate. SUPERVISOR will automatically re-initiate RECORDER the next time a RECORD statement is executed. In normal operation, the only reason to do this is to force RECORDER to create new disk files immediately after midnight.

For example, the following ODTSequence and scheduled entry could be used:

```
TT DEF + ODTS QUIT_RECORDER:  
    Record [0] ("QUIT");  
  
TT AF 0000 DAILY: DO QUIT_RECORDER
```

RECORD file names for DISK and VAR

By default, RECORDER files of type DISK are created with FILEKIND of JOBSYMBOL with MAXRECSIZE=15 words. RECORDER files of type VAR are variable length record files with BLOCKSTRUCTURE=VARIABLE and MINRECSIZE=1. VAR files were originally intended for creating files that were to be transferred to PC platforms but have been superseded by the BYTE type.

Both these files are typically called:

```
*SUPERVISOR/RECORD/<dayname>/<filenumber>
```

where <dayname> is the day of the week that the file was created e.g. MONDAY, TUESDAY etc., and <filenumber> is the RECORD file index or message class.

Two examples are shown below:

```
*SUPERVISOR/RECORD/MONDAY/3
*SUPERVISOR/RECORD/SUNDAY/53
```

If a mapped mnemonic configuration item is used, this title format is changed. If, referring to the first example above, message class 3 had been mapped to DISK and an identity of 'SYSMON' then the last two file levels are interchanged.

Example:

```
*SUPERVISOR/RECORD/SYSMON/MONDAY
```

The files are automatically placed on the disk family assigned to DL BACKUP.

New files are *not* automatically created at midnight. Although open, these RECORDER files can be manually copied if desired. If new files are required for each day – which is often a sensible idea – then scheduling a TT REC QUIT command shortly after midnight will cause the files to be CLOSED as RECORDER terminates.

SUPERVISOR will automatically re-initiate RECORDER, thereby creating new files for each day.

All RECORDER disk files have the attributes PROTECTION=SAVE and SYNCHRONIZE=TRUE which means that the files are entered into the disk directory as soon as the file is first opened; also, any logical write to the file is immediately forced as a physical write. This mechanism ensures that the files are very secure, even through halt-loads, and can be copied or accessed, without loss of data, whilst the files are still open.

RECORD file names for BYTE

RECORD files of type BYTE are Unisys stream files with attributes of FILESTRUCTURE=STREAM, EXTMODE=ASCII and MAXRECSIZE=1. Since stream files on Unisys mainframes running NX/Services are readily accessible from mapped MCP shares, stream files created by RECORDER have a '.TXT' extension.

Example:

```
*SUPERVISOR/RECORD/CRITCL/"MONDAY.TXT"
```

By default, no stream files available in an unmapped REC NAMES configuration so BYTE files can only be created with a mnemonic name and not a number.

It is not possible to view these files, using PC applications like NotePad or Microsoft Word, whilst the files are open on the mainframe.

The file must first be closed using the REC CLOSE command:

```
TT REC CLOSE CRITCL
```

Examples

```
RECORD[RSVP] (TIME(TIMEOFDAY) , MIXNO , / , RSVP) ;  
RECORD[#A] ("THIS IS A TEST STRING")  
RECORD[12] (TEXT)
```

HOTLINE program

The Metalogic HOTLINE program receives messages from one or more RECORDER clients running on either BNA or TCPIP networked systems. By default, HOTLINE expects file numbers, or messages classes, 6,7,8,9,10 and 11 to be used for port files services – though this may change if a REC_FILEMAP configuration item has been successfully processed.

To send messages to the HOTLINE program, the same OPAL syntax for recording to a disk file may be used. RECORDER will send the message to any HOTLINE program, which has requested that class of messages.

The supplied HOTLINE program can be run from any terminal. It asks the operator what hosts, and what message classes, to monitor. The operator can select message classes 6 through 11 singly, or can select ALL message classes. HOTLINE then connects to the RECORDER program on the selected hosts and requests the selected message classes. Upon receiving the messages from RECORDER, it writes the messages to the terminal. Again, if a REC_FILEMAP is used, the classes of message type available may be different.

If no HOTLINE program is currently requesting the message class, the message is discarded. If the message class is invalid, the message is discarded. Messages displayed by HOTLINE include time and host identification. If you need to identify the specific source OPAL, you should write the RECORD statement to include this information.

Operating the HOTLINE program

Operating the HOTLINE program is a simple matter of running the program and answering a couple of questions. From CANDE, enter:

```
RUN *METALOGIC/SUPERVISOR/HOTLINE
```

HOTLINE will now ask which hostnames should be monitored (note that the local hostname is **always** monitored). Enter the names of any other hosts you want to monitor, one name per transmission. Then transmit a space to terminate the list.

```
HOTLINE version 45.450.005 [Compiled 15:44:51, 06/07/1999]  
Connecting to local host CPMCP1  
You may connect to another 10 hosts  
Enter hostname, '+' for default, ' ' to stop
```

Next, HOTLINE will check with RECORDER to see if there any local port file mappings and then ask for a list of message types to be monitored:

```
One moment, asking CPMCP1 RECORDER for the names of message
types
```

```
Please select the messages to see
```

```
Enter ALL or some of:
```

```
    RSVP, ALERT, DSED, 9, 10, 11, 47
```

Note that any port number that does not have a mapped ID just returns the number. In this example, the user asked for all message types by entering 'ALL':

```
ALL
```

```
OK
```

```
CPMCP1 Type mask set to decimal 0 ; hex 7FFFFFFFFFFFFFFF
```

```
Enter 'QUIT' to terminate HOTLINE
```

```
Enter 'STATUS' for status
```

```
Enter 'HELP' for more commands
```

HOTLINE will then enter monitoring mode, displaying each message on the terminal as the message arrives. Some commands are available to provide additional status and run-time information and HOTLINE will show these once the message classes have been selected, use the HELP command for more information.

Multiple copies of the HOTLINE program can be running at the same time, either on the same host or on different hosts. If more than one copy of HOTLINE requests the same message class, all copies of HOTLINE receive copies of all messages in that message class.

Much of the work in both HOTLINE and RECORDER recently has been to provide support for NAP systems. In particular, HOTLINE has additional functionality for NAP, which is of no relevance to other customer sites and has NOT been included in this document.

Customising RECORDER and HOTLINE

The source files for RECORDER and HOTLINE are supplied on the METALOGIC release media. You can modify or rewrite these programs for any purpose whatever. You may wish to keep the following considerations in mind:

If you modify RECORDER but still wish to use the supplied HOTLINE program, you must maintain the HOTLINE interface.

RECORDER must run in the same system as SUPERVISOR; if you need functions which are better performed on a remote host, or if you need functions performed in one place for an entire network, you should probably consider modifying the HOTLINE program instead of RECORDER.

If you need a feature in Recorder which would tempt you to compile your own version you should contact us first. We are very amenable to customer requests to enhance the software and having your own version of Recorder will complicate subsequent software installations.

Only one RECORDER can be attached to each SUPERVISOR. Many different HOTLINE programs – as well as multiple copies of one HOTLINE program – can be attached to a single RECORDER. This is another reason for choosing to modify HOTLINE rather than RECORDER. You can continue using the supplied HOTLINE program intact, while also running a completely different HOTLINE program, which receives different message classes and takes different actions.

RECORDER connects to HOTLINE by offering a port sub-file. See the source files for the appropriate declarations. RECORDER attempts to keep one port sub-file offered at all times, unless the maximum number of connections has been reached. HOTLINE attempts to open a port sub-file to RECORDER on each host selected:

```
CANDIDATE RECORDERHOTLINE
CANDIDATE:  FILENAME = RECORDERHOTLINE
MY HOST      = POWEREDGE      MY HOST GROUP    = POWEREDGE
APPLICATION GROUP =          MY USERCODE      =
REQUESTED CHAR SET = EBCDIC    TRANSLATE        = NOTRANS
OFFER TYPE    = NO OFFER      MATCHING STATUS   = NOT OFFERED
MY NAME       = RECORDER      YOUR NAME         = HOTLINE
MY CODEFILE = *METALOGIC/SUPERVISOR/RECORDER ON DISK
USING PROCESS = 18676/18676    INTNAME            = HOTLINE
SUPPORT INDEX = 1             YOUR HOST           =
YOUR HOST GROUP =
REQUESTED YOUR USERCODE =
AVAILABLE ONLY = -            IGNORE DIRECTORY    -
SECURITY TYPE  = PUBLIC
```

RECORDER keeps track of connection open and closes, regularly displaying changes in port sub-file states.

After the RECORDER – HOTLINE port is open, HOTLINE must write a one-word record to the port sub-file. The one word is a bit mask of the message classes requested. HOTLINE can change its request at any time simply by writing a new mask to the port sub-file. This convention limits message classes to a maximum of 47 unless coding is done to change the convention.

RECORDER timestamps the message before sending it to HOTLINE; the original text will be preceded by the time in the form **hh:mm:ss.s**, and a space (11 characters total). A HOTLINE program connected to multiple hosts should keep track of which sub-file is connected to which host. If the HOTLINE needs to identify the originating OPAL, you should write the **RECORD** statements to include the required identification in the text.

COMS and HOTLINE

In COMS UTILITY, it is possible to set up a window for the exclusive use of HOTLINE by specifying METALOGIC/SUPERVISOR/HOTLINE (or the file title you have given your codefile) as the remote file PROGRAM for a WINDOW (normally called HOTLINE) using a remote file interface.

This means that HOTLINE can be accessed by the command:

```
?ON HOTLINE
```

In order to run HOTLINE without the prompts for message classes and hostnames, especially where the HOTLINE station is a printer device, the following file-equations are necessary:

```
FILE HOST1 = PRODUCTION;FILE HOST2 = LINC;
```

If any host is equated, no questions about hosts will be asked. If not connecting to any other hosts and FILE HOST0=<myhostname>, no questions will be asked about hostnames.

The task attribute TASKSTRING should be used to select the message classes to be handled. This is done by specifying a class "list", each class being separated by a comma:

```
TASKSTRING="6,8,9"
```

The current implementation only permits a number list in TASKSTRING but it is anticipated that a mnemonic list will be implemented soon.

Further, the COMS HOTLINE window declaration can be modified to include the file equations or, if run remotely, normal run-time file equations can be used. Alternatively, to make the settings permanent, a WFL MODIFY of the HOTLINE codefile can be performed.

For example:

```
WFL MODIFY *METALOGIC/SUPERVISOR/HOTLINE;  
TASKSTRING= "6,7,8,9,11,17";FILE HOST1 = PRODUCTION;
```

HOTLINE_MSG library entryptpoint

RECORDER now freezes as a library and exports the procedure:

```
PROCEDURE HOTLINE_MSG(TEXT);  
EBCDIC ARRAY TEXT[*];
```

Example program using Hotline_Msg:

```
00001000$Set Level 2
00002000Procedure Record(Param) ;
00003000Array Param[*] ;
00004000Begin
00005000Library RecorderLib(Libaccess=ByFunction) ;
00006000Procedure Hotline_Msg(Text) ;
00007000          Ebcdic Array Text[*] ;
00008000          Library RecorderLib ;
00009000Ebcdic Array EParam[0:255] ;
00010000Pointer PIn,Pout ;
00011000Pout:=EParam ;
00012000If PIn:=Pointer(Param) = 48"00" For 1 Then
00013000Begin
00014000    Display("Requires a string parameter") ;
00015000    Myself.Status:=Value(Terminated) ;
00016000End ;
00017000If PIn Neq "[" For 1 Then
00018000    Replace Pout:Eparam By "[Debug]" ;
00019000Replace Pout By PIn Until = 48"00",48"00" ;
00020000Hotline_Msg(EParam) ;
00021000End.
```

This procedure allows a user program to pass a message to RECORDER as if an OPAL RECORD statement had been executed. The TEXT parameter is used to hold the message, which may start with a control sequence, to denote the mnemonic or file number to use.

For example, if the TEXT parameter held:

```
"[WAIT]THIS IS A TEST"
```

This equivalent to, from an ODTSequence:

```
RECORD[WAIT] ("THIS IS A TEST")
```

Similarly, to route a message to RECORD [6]:

```
"[6]YET ANOTHER TEST"
```

If the prefix does not map onto a valid 'REC NAME' or number, then the prefix is not stripped from the text and the default record value is used.

If the REC NAME mnemonic DEFLIB has been previously defined then that name is used for a default RECORD i.e. no control prefix is present in TEXT. If DEFLIB is not defined then RECORD[0] is used as the default

A TT REC QUIT command will return a warning if user programs are currently linked to Recorder.

A subsequent TT REC QUIT will force RECORDER to quit, force terminating any tasks that are currently linked.

A TT REC VERSION will show how many tasks are linked and also the mix number

of RECORDER. A <recorder mix number>Y command will list all of the linked tasks. The TT REC VERSION command resets the 'pending quit' flag so that if tasks are still linked it will again require two QUIT commands to terminate.

Examples of using RECORD

First, run the HOTLINE utility from CANDE on the local host (in this case COURSE2MCP) and ask for DEBUG message classes to be detected. Once running, a RECORD[DEBUG] was done from Supervisor. Input to Hotline is is light blue:

```
RUN *METALOGIC/SUPERVISOR/HOTLINE
#
#RUNNING 1861
#?
HOTLINE version 51.510.002 [Compiled 17:34:59, 08/04/2008]
You may connect to another 9 hosts
Please enter a hostname, ' ' to stop
blank transmitted
Connecting to local host COURSE2MCP
Enter 'QUIT' to terminate HOTLINE
Enter 'STATUS' for status
Enter 'HELP' for more commands
COURSE2MCP 11:12:25 HOTLINE OPENED
One moment, asking COURSE2MCP RECORDER for the names of message types
Please select the messages to see
Enter ALL or some of:
ALL, 6, 7, 8, 9, 10, 11, META_RECORD, META_STATUS, META_ALERT, DEBUG, META_MIN
META_FULL, META_WARN, AC, CONTAIN, TEST
DEBUG
Classes checked OK
COURSE2MCP 11:14:47 [DEBUG]This is a test
```

Assuming that it is necessary to capture and record all ACCEPT waiting entries to a HOTLINE station. The following is an example SITUation that might handle this:

```
TT DEFINE + SITUATION WAITING(MX=WAITING) :
    "ACCEPT:" ISIN RSVP
SITUATION DEFINITION ENTERED
```

Next, an ODTSequence is needed to perform the RECORD, similar to REC_MSG:

```
TT DEFINE + ODTSEQUENCE WAITING(MX) :
    RECORD[6] ("ACCEPT DETECTED: ",MIXNO,,RSVP) ;
ODT SEQUENCE DEFINITION ENTERED
```

SUPERVISOR Operation

When SUPERVISOR enters the mix, it first acts to ensure that its program name corresponds to the METALOGIC standard, which in this case means that its NAME should be of the form:

<OPTIONAL PREFIX> METALOGIC/SUPERVISOR

<OPTIONAL PREFIX> is an optional site-specified prefix to the file TITLE of the code file. This is needed to correctly construct the names of other METALOGIC Utilities, which SUPERVISOR may run. If the TITLE is not in this form, the initialisation will abort with the message:

SUPERVISOR MUST HAVE 'METALOGIC' AS ONE LEVEL OF ITS NAME

Whenever SUPERVISOR is initiated manually, it should be done with a primitive run (that is, ??RUN), rather than with a normal run as the second form creates a WFL job. This is undesirable because at the next Halt Load there will be two SUPERVISORs, one initiated by the MCP because it is the AI list and one because the job restarts. It is also marginally inefficient, because the WFL job is an extra-unwarranted overhead. SUPERVISOR ensures that a primitive run was performed by checking if its JOBNUMBER task attribute differs from its MIXNUMBER task attribute. If SUPERVISOR determines this is the case, the current job will terminate itself and simulate a primitive run of the SUPERVISOR codefile. The following message will be displayed:

SUPERVISOR is being re-run as an independent task

The next action taken is to initialise a primary queue, which marks SUPERVISOR as an MCS. If the name of the code file is not declared to the system as a valid MCS, SUPERVISOR will add itself as an MCS.

If the system option AUTODC is reset and one or more datacom tasks are not already running, then SUPERVISOR will set AUTODC, open a primary queue and then reset AUTODC. To be able to do this the SUPERVISOR code file must be a privileged codefile (i.e. MP..+PU) Should SUPERVISOR already be running, the MCP will not permit a second copy to initialise its primary queue.

In this case SUPERVISOR will abort with the message:

ONLY ONE SUPERVISOR ALLOWED

Having become an MCS, SUPERVISOR procures one of the system schedule stations, and then attaches itself to CONTROLLER via the SETUPINTERCOM intrinsic. As the next step, now that SUPERVISOR has privileged status, it can call the MCP GETSTATUS intrinsic. GETSTATUS is used to establish the environment, type of machine, and so on. Access to the JOBDESC file is verified at this point.

Should SUPERVISOR be unable to locate or read the JOBDESC file, it will abort with the message:

CANNOT LOCATE DL JOBS FAMILY

SUPERVISOR will link to both the MAGUS and GENERALSUPPORT libraries and

attempts to locate its SCHEDULE file. This file should be located on the DL JOBS family. If it is not, an attempt is made to reload a backup copy from the BACKUPFAMILY. If no backup copy exists, a new SCHEDULE is created.

In the normal case, a SCHEDULE is located. This file is then validated in a very thorough process, which includes both address checking and checksum on every record.

When successfully completed, a message is displayed giving the amount of space left in the file:

SCHEDULE IS 96% AVAILABLE (610 SEGMENTS IN USE)

Should errors occur, either I/O errors or structural errors, SUPERVISOR will discard the SCHEDULE, changing its title to OLDSCHEDULE, and attempt to reload the backup copy. The validation process is then repeated for the backup copy. Should that validation also fail, this SCHEDULE is discarded and a new file is created; SUPERVISOR will also generate a program dump and a descriptive message. Any job summaries and program dumps should be retained for future investigation by Metalogic personnel.

Messages indicating inconsistencies have the general form:

SCHEDULE INCONSISTENCY:ERROR <number> <message>

Whenever a new SCHEDULE is created, it too undergoes the validation process above.

If this validation fails, SUPERVISOR will abort with the message:

COULD NOT VALIDATE NEW SCHEDULE

SUPERVISOR always opens the SCHEDULE file with the EXCLUSIVE file attribute set TRUE. In addition, the FILEKIND is set to RECOVERYFILE, SECURITYTYPE to PRIVATE, and in general any possible step, which makes access difficult for other programs, is taken. METALOGIC does not support access to this file by programs other than SUPERVISOR.

I/O errors during SCHEDULE creation can cause various aborts at this stage and usually indicate that the DL JOBS family pack is unreliable.

Whenever a new SCHEDULE is created, SUPERVISOR outputs a message:

A NEW SCHEDULE WAS CREATED

and then searches for the secondary backup file specified by the USE FILE FOR REBUILD setting.

If this has not been set, SUPERVISOR will look for the default backup file called:

(<usercode>) SAVED/SCHEDULE/DEFAULTS

which it hopes to find on the BACKUPFAMILY. If found, the file will be used as default input to assist the creation of the new SCHEDULE file. If this action is taken, a SAVE SCHEDULE command will be scheduled after the completion of the

initialisation process.

Any sessions allocated to dependent tasks by a previous run are released at this point. Restart information for WHENs and DOs is examined. Any sessions allocated for these are released with a log-off reason of "Restart of MCS". Wherever possible, these WHENs and DOs are restarted with the following exceptions: typed DOs with parameters cannot be restarted unless they are of type MSG, also any WHEN ... DISPLAYS initiated by Supervisor Windows or REMOTESPOs will not be restarted

SUPERVISOR then checks MCP level for compatibility and logs his own version in his JOB LOG.

If the queue of the USE QUEUE command does not exist, it sends the system command:

```
MQ <queue number> ML 1, DEFAULTS(PRIORITY = 50),  
FAMILY DISK=<halt-load family> OTHERWISE <system family>
```

If the HARDCOPY Option was set, SUPERVISOR will restart SYSTEM/HARDCOPY.

SUPERVISOR then simulates the following system commands:

First, SUPERVISOR adds its codefile name into the AI (Automatic Initiator) list and shuffles the existing entries so that it is executed first;

Second, SUPERVISOR will LP its own job number to prevent accidental DS.

During initialisation, SUPERVISOR must decide what sort of situation it is in – is this a Halt Load, COLDSTART, a simple restart after a QUIT, or just a manual run? SUPERVISOR is normally initiated by the MCP after a Halt Load, but may also be initiated with an explicit system command. The difference is important, as it must decide whether to perform any activities scheduled with the AFTER HL variant.

If there has been a Halt Load, the AFTERHALTLOAD process is invoked.

SUPERVISOR knows there has been a Halt Load by checking elapsed time against the MCP's value for time since the last Halt Load (maintained by TIME intrinsic 14).

If SUPERVISOR has started within one hour of a coldstart, it will do

AF+COLDSTART scheduled activities

If SUPERVISOR has started within 30 minutes of a Halt Load it will do

AF+HL activities

If SUPERVISOR decides to perform either of these actions, it will at this point set the system emergency interrupt (HS system command).

Halt Load action includes transfer to the user-specified Halt Load sequence, which is done by running the task "AFTER HALTLOAD", or in the case of a COLDSTART, "AFTER COLDSTART". "AFTER HALTLOAD" first performs any scheduled activities scheduled by the AFTER HL command. If the Halt Load reason was a Cold Start, activities scheduled by the "AFTER COLDSTART" command are performed before the Halt Load activities.

Once the "AFTER HALTLOAD" process has completed, SUPERVISOR will invoke any activities that have been configured by the AFTER RESTART command. Typically, this feature is of high importance if the SUPERVISOR option NOWHENRESTART was set. In this case, the AFTER RESTART should include an OPAL script to handle the controlled invocation of any required WHENs.

Next, if NOWHENRESTART is reset, SUPERVISOR will automatically re-initiate any WHENs or DOs the last time that SUPERVISOR was running. However, any WHEN..DISPLAY combinations will be discarded unless they were originally run from a system ODT.

During this phase, if the TRIM module is active, SUPERVISOR will run the LOGREADER utility to scan the SUMLOG(s) for any missed tape activity. LOGREADER communicate with SUPERVISOR to determine the timestamp of the last know update of the tape database; this allows LOGREADER to keep access to the SUMLOG(s) to an absolute minimum.

SUPERVISOR has the possibility of backing up the USERDATAFILE. It now checks that the USERDATAFILE is resident on the DL USERDATA family. If not, it attempts to reload its backup copy (simulating the effects of an RL USERDATA command). It then changes the password of its own usercode, if necessary creating the usercode, and setting desired values for the USERDATA attributes.

At this point, SUPERVISOR has initialised, and to confirm its success it outputs a WS response to each ODT.

Tailoring SUPERVISOR

It is possible to "tailor" the SUPERVISOR codefile by modifying certain task attributes by using the MODIFY command or using SYSTEM/BINDER. However, in most cases, the standard released codefile needs no changes since SUPERVISOR's run-time environment is highly configurable.

The following section has been provided for completeness.

Task Attributes

The site can to some extent modify the task attributes of the external tasks fired off by SUPERVISOR. For sites, which wish to direct output from all stacks processed from SUPERVISOR to remote printers (of whatever kind), it is possible to set the CANDEDESTNAME USERDATA attribute (with MAKEUSER) for SUPERVISOR's own usercode, as DEFINEd by the USE USER command. The value of the attribute is checked dynamically just prior to the initiation of each task, to ensure any MAKEUSER action takes effect immediately. If no value is supplied for the attribute, the value of DESTNAME defaults to SITE.

To enable site control of the value of DECLAREDPRIORITY for SUPERVISOR's processed tasks and jobs, it uses the value of the USERDATA attribute PRIORITY obtained from the current USERDATAFILE. If no value is set, tasks will run at priority 50.

Whenever DL BACKUP is changed, SUPERVISOR's BACKUPFAMILY task attribute will automatically be changed to reflect the system DL BACKUP setting. Any program initiated via an ODTSequence inherits SUPERVISOR's BACKUPFAMILY task attribute, as determined at the task initiation.

Renaming files

The code files produced for the running system are named METALOGIC/=. There are restrictions on the names allowed: (1) METALOGIC must appear somewhere in the name, and (2) the remainder of the title after METALOGIC may not be changed. Thus

```
CHANGE METALOGIC/= TO SYSTEM/METALOGIC/=
```

is fine, but

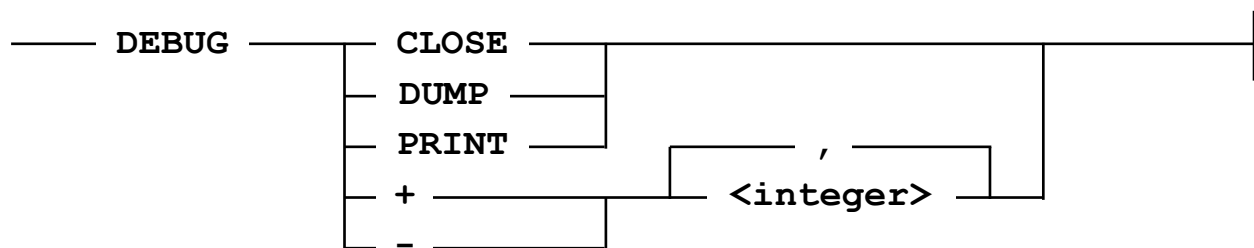
```
CHANGE METALOGIC/SUPERVISOR/= TO META/=
```

fails on the first count, and

```
CHANGE METALOGIC/SUPERVISOR/= TO METALOGIC/SUPERVISOR/NEW/=
```

fails on the second count.

DEBUG Command



The DEBUG command is used to control debugging facilities within SUPERVISOR. With no qualification, DEBUG returns a list of numeric debug options, each with a one-line explanation. The DEBUG +, and DEBUG - options allow "bits" to be set or reset respectively.

The DEBUG DUMP command takes a program dump and then executes a DEBUG PRINT command. The DEBUG PRINT command closes SUPERVISOR's TASKFILE and enters a print request for it. This allows printing the job summary and debug information in the TASKFILE without having SUPERVISOR go to EOJ. The DEBUG CLOSE has the same effect as DEBUG PRINT.

The DEBUG command only exists if the SUPERVISOR compile-time option DEBUG has been set though it would be very unusual for DEBUG to be unavailable in released software. The functions of any <bit number> are described in the response and may change without notice from release to release.

In normal use all options should be reset.

Examples

```
TT DEBUG
TT DEBUG + 1,6,19
TT DEBUG CLOSE
TT DEBUG - 11,12,15,46
```

Debugging OPAL programs

The DEBUG option of the OPAL execution commands (Evaluate, WHEN, DO etc.) provides a printed trace of the execution. It is invoked by putting **&DEBUG** after the command keyword, e.g.

```
TT EV & DEBUG <program id>
```

When using this option it is important to obtain a code listing. This is accomplished by modifying the SITUation with the DEF command. The following SITUation has been modified to obtain a code listing:

```
TT DEFINE + SITUation iii_MIX_FINDER_X (MX) SET LIST CODE:
"METALOGIC" ISIN NAME
```

To print the trace

```
TT DEBUG CLOSE
```

should be entered so that the printer backup file (*BD/000nnnn/TASKFILE) will be closed and released. DEBUG can also be used with DO, /, WHEN and ONCE commands.

Fault Handling

SUPERVISOR has two types of fault handling, one for internal faults caused by malfunctions within SUPERVISOR itself and one for "cleaning up" when the pseudo-code generated by the OPAL Compiler causes a fault. The first type of fault is always reportable and generally indicates an error within SUPERVISOR. The second type usually indicates a fault in an OPAL program, and is normally a user programming error.

The messages from SUPERVISOR indicate which type it is, i.e.

```
FAULT TERMINATION WITHIN SITUATION
FAULT TERMINATION WITHIN ODTSEQUENCE
FAULT TERMINATION WITHIN DISPLAY
FAULT TERMINATION WITHIN SUPERVISOR
```

In general, SUPERVISOR forces a divide by zero when it detects certain non-obvious user errors, such as attempting to make strings which are too long, errors from GETSTATUS calls and the like. They are usually accompanied by an explanatory display and may also be distinguished by looking at the program dump (which is always taken), as the top two words in the user portion of the stack contain

a word of EBCDIC characters giving some indication of the problem, and a zero.

It cannot be guaranteed that all user errors are successfully distinguished from internal errors. Therefore to aid both the user and METALOGIC support staff in difficult cases, a SUPERVISOR compiled with the compile-time option DEBUG will retry the failing code with a special run time DEBUG trace option. The listing will first show the definition of the failing program, then a trace of the failing instructions and their operands, and will end with a normal program dump. The DEBUG option 23 or the SET CODE compiler option can be used for further detailed analysis.

Explanatory messages for most errors are DISPLAYed from SUPERVISOR's stack and appear in the log and the job summary.

If the error is recognisably an internal fault, SUPERVISOR will detect errors of the same type (zero divide, segmented array, etc.) from the same internal line number and for second and subsequent faults at that point issue a message like

```
PROGRAMDUMP NUMBER 5 FROM SEQUENCE 81234567 NOT TAKEN
```

To force all dumps to be taken, DEBUG option 4 should be set. The memory of prior faults extends only to beginning of the current run, that is, since the BOJ of SUPERVISOR.

If you are going to use a datacom station to communicate with SUPERVISOR then your usercode and/or accesscode must have the necessary privileges to access system functions. System ODTs do not normally have any restrictions imposed upon them but with SUPERVISOR, sites have the option of validating input. If a SLed Library METASECURITYLIB exists then SUPERVISOR calls the entry point VALIDINPUT. This Boolean function is expected to return true if input is valid and false if invalid. The user is expected to generate an error message if invalid.

A file is provided on the software release tape:

EXAMPLES/SUPERVISOR/SECURITYLIB

and this shows a very simple example of a user library to do this checking.

If SUPERVISOR detects the SL-ed Library METASECURITYLIB, it will also do password checking on SHIFT BEGIN Commands.

If you are in any doubt about your usercode or accesscode security level, please contact your System Administrator before trying out any of the SUPERVISOR commands.

Release Media

Below is a listing and short description of the files contained within the standard SUPERVISOR wrapped container:

***METAKEYS/<Customer id><Version>**

Licence keys file.

***OBJECT/META/INSTALL**

***OBJECT/META/INSTALL/SCREENLIB**

Install utility and support library.

***METALOGIC/MAGUS**

Library of advanced routines, key validation.

***METANOTES/META/INSTALL**

***METANOTES/MAGUS**

***METANOTES/SUPERVISOR**

***METANOTES/SUPERVISOR/NAPINTERFACE**

***METANOTES/OPAL**

***METANONE/OPAL/COMPILER**

***METANOTES/OPAL/GSTABLEGEN**

***METANOTES/SUPERVISOR/PRINTLABEL**

***METANOTES/SUPERVISOR/RECORDER**

***METANOTES/SUPERVISOR/TTINTERFACE**

***METANOTES/MAILLIB**

Description of latest changes to all relevant software items.

***METALOGIC/SUPERVISOR**

SUPERVISOR codefile.

***METALOGIC/SUPERVISOR/RECORDER**

Recorder codefile.

***OBJECT/OPAL/BINDLIB**

Library codefile need for compiles, bound into
METALOGIC/OPALTAPELIB and
OBJECT/SUPERVISOR/OPAL.

***METALOGIC/SUPERVISOR/HOTLINE**

Simple HOTLINE utility.

***SYSTEM/OPAL**

Standalone OPAL compiler.

***METALOGIC/SUPERVISOR/TTINTERFACE**

MARC TT Directive.

***METALOGIC/SUPERVISOR/NAPINTERFACE**

Interface library for NAP.

(METASOFT) SUPERVISOR/NAPINTERFACE

Source for interface library for NAP.

(METASOFT) NAPCOMPILE/=

Tools to help recompile NAPINTERFACE.

(METASOFT) SUPERVISOR/ASSIGNCHARGE

Source to allow tailored charge codes for Supervisor tasks.

(METASOFT) MEMO/=

Memo files describing above.

(METASOFT) OPAL/USERFNLIB

Example source file for OPALUSERLIB library which used the OPAL function USERFN. Requires SL.

***METALOGIC/OPAL/USERFNLIB**

Compiled code for above example.

(METASOFT) SUPERVISOR/RECORDER

DCALGOL source for RECORDER codefile.

(METASOFT) SUPERVISOR/HOTLINE

DCALGOL source for HOTLINE utility.

(METASOFT) SUPERVISOR

DCALGOL source for SUPERVISOR program.

(METASOFT) SUPERVISOR/TTINTERFACE

DCALGOL source for TTINTERFACE directive.

(METASOFT) SUPERVISOR/LOGREADER

DCALGOL source for LOGREADER utility.

(METASOFT) SUPERVISOR/TABLEGEN

DCALGOL source for SUPERVISOR reserved word table generator.

(METASOFT) SUPERVISOR/MSGS/ERRORGEN

DCALGOL source for SUPERVISOR run-time messages.

(METASOFT) SUPERVISOR/COMPILE

WFL job to compile and bind SUPERVISOR.

(METASOFT) INCLUDE/MCP

Include file used by for SUPERVISOR and OPAL compiles.

***OBJECT/SUPERVISOR,**

Host SUPERVISOR codefile prior to final bind.

***OBJECT/SUPERVISOR/OPAL**

Bindable OPAL codefile unit bound into OBJECT/
SUPERVISOR.

(METASOFT) METATAPELIB/DASDL

Base DASDL source for METATAPELIB database compile.

(METASOFT) OPALS/SUPERVISOR/TAPEDB

Example OPALs for Tape Library Database support.

(METASOFT) OPALS/SUPERVISOR/COURSE

OPALS used on the Metalogic OPAL Course.

(METASOFT) OPALS/SUPERVISOR/EXAMPLES

Examples of various OPAL programming techniques.

(METASOFT) OPALS/SUPERVISOR/GETTINGSTARTED

Example OPALs as described in [Advanced Opal](#).

***METALOGIC/SUPERVISOR/TAPELIBUPDATER**

SUPERVISOR controlled codefile which routes tape information and commands to the METATAPELIB database.

(METASOFT) METATAPELIB/OPALTAPELIB

Source for OPALTAPELIB library compiled against customer DESCRIPTION file by INSTALL. Library automatically SLED by INSTALL.

***METALOGIC/SUPERVISOR/LOGREADER**

LOGREADER utility for extracting tape information from SUMLOGS and updates METATAPELIB database, if necessary.

***OBJECT/TRIM**

Tape Library support utility allowing maintenance of tape expiry and retention rules.

***METALOGIC/METATAPELIB/SCREENLIB/TRIM**

Source support library for TRIM remote screens.

(METASOFT) DESCRIPTION/METATAPELIB/OLD

Old METATAPELIB description file for Tape Library install may be necessary where customer DESCRIPTION file is lost.

(METASOFT) METATAPELIB/SCREENLIB/TRIM

TRIM support utility required for screen I/O.

***METALOGIC/SUPERVISOR/WAITWATCHER**

WAITWATCHER codefile - bundled with SUPERVISOR – has scaled down SUPERVISOR capabilities.

Advanced Opal examples

Many of the following OPALs were specifically written to address real-life customer problems and range in complexity. Some of the examples show here use advanced OPAL programming techniques – don't let this put you off if you are just starting to get acquainted with the software. You can always come back to these examples at a later stage. It is still worthwhile, though, trying them out even if you don't completely understand how they work.

These examples will already have been loaded if you have been following [Getting Started](#) and ENTERed from the file OPALS/SUPERVISOR/GETTINGSTARTED. If you have not entered from the file, each example describes how to enter the code for it alone.

Example: COM HL_INFO

This example keeps a short history of system Halt/Loads.

To load it

```
ENTER COM HL_INFO FROM OPALS/SUPERVISOR/GETTINGSTARTED
```

The first time the command is entered from a Supervisor window it will update the schedule so that all subsequent halt/loads are recorded.

Once some data has been recorded (after some Halt/Loads) The command will produce a report like:

```
HL_INFO
=====
HALT-LOAD STATISTICS on DELL MCP
=====
2008/06/30 10:41:43 MCP 12.0 (53.180.1258) HL REQUEST
2008/06/26 11:51:29 MCP 12.0 (53.180.1258) HL REQUEST
2008/06/24 12:29:57 MCP 12.0 (53.180.1258) CONSOLE LOAD
2008/06/24 11:51:13 MCP 12.0 (53.180.1258) CONSOLE LOAD
2008/06/24 11:24:31 MCP 12.0 (53.180.1258) HL REQUEST
2008/06/20 14:34:26 MCP 12.0 (53.180.1258) CM REQUEST
```

Example: COM SUP_INFO

This example is similar to HL_INFO but keeps a short history of Supervisor restarts.

To load it

```
ENTER COM SUP_INFO FROM OPALS/SUPERVISOR/GETTINGSTARTED
```

The first time the command is entered from a Supervisor window it will update the schedule so that all subsequent Restarts are recorded.

Once some data has been recorded (after some Restarts) The command will produce a report like:

SUP_INFO									
Supervisor Statistics on DELL MCP									
=====									
2008/06/30	10:46:29	SUPERVISOR	Version	53.530.65	10:22:48	on	27/06/08		
		Attributes	Version	53.530.21	14:52:54	on	12/06/08		
		OPAL	Version	53.530.23	10:11:03	on	27/06/08		
		MAGUS	Version	53.530.10					
2008/06/27	11:19:45	SUPERVISOR	Version	53.530.65	10:22:48	on	27/06/08		
		Attributes	Version	53.530.21	14:52:54	on	12/06/08		
		OPAL	Version	53.530.23	10:11:03	on	27/06/08		
		MAGUS	Version	53.530.10					
2008/06/26	11:55:23	SUPERVISOR	Version	53.530.63	10:27:49	on	26/06/08		
		Attributes	Version	53.530.21	14:52:54	on	12/06/08		
		OPAL	Version	53.530.22	13:19:44	on	25/06/08		
		MAGUS	Version	53.530.10					
2008/06/26	10:33:16	SUPERVISOR	Version	53.530.63	10:27:49	on	26/06/08		
		Attributes	Version	53.530.21	14:52:54	on	12/06/08		
		OPAL	Version	53.530.22	13:19:44	on	25/06/08		
		MAGUS	Version	53.530.10					
2008/06/26	10:12:21	SUPERVISOR	Version	53.530.63	10:07:08	on	26/06/08		
		Attributes	Version	53.530.21	14:52:54	on	12/06/08		
		OPAL	Version	53.530.22	13:19:44	on	25/06/08		
Window S/1 at DELL MCP									

Example: COM DBS

This example is intended to replace the MCP command DBS.

ENTER COM DBS FROM OPALS/SUPERVISOR/GETTINGSTARTED

For each database in the mix it will return both statistics and a list of users.

DBS				
Mix Users 1 Active Database				
4105	6	(BOB)ACDB1		
Allowed	30000	InUseCore	888	
MaxInUse	888	MaxBuff	2	
OverlayGoal ..	5.000%	OverlayRate ..	0.000%	
Transactions .	3	ForcedOverlays	0	
NormalAudits .	0	Syncpoints ...	0	
Programs in Transaction state:				
T0004086 U 50 (BOB)OBJECT/AC/DB1/MOD/3 ON DEV				
T0004084 U 50 (BOB)OBJECT/AC/DB1/MOD/2 ON DEV				
T0004081 U 50 (BOB)OBJECT/AC/DB1/MOD/1 ON DEV				
Programs not in Transaction state:				
0004096 I 50 (BOB)OBJECT/AC/DB1/INQ/3 ON DEV				
0004095 I 50 (BOB)OBJECT/AC/DB1/INQ/2 ON DEV				
0004094 I 50 (BOB)OBJECT/AC/DB1/INQ/1 ON DEV				

As with the MCP command it will allow filtering by Usercode or Name:

DBS NAME =DB1

DBS USER BOB

Example: COM Y

This example is intended to replace the MCP command Y.

ENTER DEF Y FROM OPALS/SUPERVISOR/GETTINGSTARTED

This will define an ODTs and a COM named Y. (The ODTs is used to return a hierarchical list of linked programs if the mixnumber passed is a library). It returns

much more information than the MCP Y command.

```
3619 Y
----- ACTIVE ENTRY 3961/3961 at 12:23:28 -----
Name      : *METALOGIC/MAGUS
CodeFile  : *METALOGIC/MAGUS ON DISK
           16:33:49,12/06/08
User      : In queue : 0 Priority :50
Charge    :
State     : Frozen
Display   : MAGUS:COMPILED AT 16:34:00 ON 12 JUN 08
Origin    : Unit 0
          PPed  CPed
This Library is being used by by 5 programs
4183  *METALOGIC/MAILLIB
Has 1 User:
      3953  *METALOGIC/SUPERVISOR
4047  *METALOGIC/FLEX/LIBRARY
Has 1 User:
      3953  *METALOGIC/SUPERVISOR
3976  *METALOGIC/OPAL TAPELIB/SLAVE
Has 3 Users:
      4198  (SUPERVISOR)SUPERVISOR/TAPELIBUPDATER
      3962  *METALOGIC/TAPEMANAGER
Has 1 User:
      21    MCP
      3953  *METALOGIC/SUPERVISOR
3954  *METALOGIC/SUPERVISOR/WAITWATCHER
3953  *METALOGIC/SUPERVISOR

Times    :      Process    = 00:00:01      Rate: 0%
           IO              = 00:00:13      Rate: 0%
           ReadyQ         = 00:00:03      Rate: 0%
           InitPbit       = 00:00:00
           OtherPbit      = 00:00:00      139 Operations % of proc Time= 3%
           Elapsed        = 25:41:05      0 Operations  % of proc Time= 0%
           ACTIVE         = 25:37:52

Core     :      Total      Save
Stack    29393      4019  <<<
Code     11597      292
Total    40990      4311

2 files for mix 3961
Kind  Use  Reads  Writes  Trans  Time  File Title
Pack  I/O      8    163    223   0:00:02 *METALOGIC/MAILLIB/LOG ON CDIMAGE.
Pack  I/O     32    743    989   0:00:07 *METALOGIC/SUPERVISOR/LOG ON CDIMAG
```

Example: COM NETWORK

This example combines various NW commands into one.

```
ENTER COM NETWORK FROM OPALS/SUPERVISOR/GETTINGSTARTED
```

Example

```
NETWORK
TCPIPHOSTNAME = DELLMCP.METALOG.COM
TCPIP IS CURRENTLY NETWORKING(IPV4-ONLY),
  RIP IS CURRENTLY ENABLED/RUNNING,
  TCPIP SECURITY IS CURRENTLY RUNNING,
  SSL IS CURRENTLY TERMINATED,
  IPSEC IS CURRENTLY DISABLED/NOT RUNNING
TCPIP IDENTITY

  IPADDRESS 192.168.22.1 / 255.255.255.0
  NP-DEVICE 210, LINEID 0,
  PHYSICAL ADDRESS HEX 08000B002100
  AUTOCONFIGURATION FALSE,
  DUPLICATEADDRESSDETECTIONTRANSMITS 0,

  IPADDRESS 10.0.0.16 / 255.255.255.0
  NP-DEVICE 210, LINEID 2,
  PHYSICAL ADDRESS HEX 000FB58A6F4E
  AUTOCONFIGURATION FALSE,
  DUPLICATEADDRESSDETECTIONTRANSMITS 1

CNS is available
NEXT INIT/INFO FILE WILL BE DELLMCP/INIT/CNS ON DEV.

  DNS Resolver Vrs 53.180.007 as DSS "DNSERVICE"
  Presently there are
  6 Linked library users
  1 TCP providers connected
  Domain name: metalog.com.
  Default name servers are:
  10.0.0.11,
  10.0.0.12

NAMESERVICE is available
NEXT INIT/INFO FILE WILL BE *NULL.

  BNA is available
NEXT INIT/INFO FILE WILL BE DELLMCP/INIT/BNA ON DEV.

NETWORK INITIALIZATION FILE IS
  DELLMCP/INIT/BNA ON DEV
NEXT NETWORK INITIALIZATION FILE WILL BE
  DELLMCP/INIT/BNA ON DEV
CURRENT NETWORK PHASE IS NODE OPERATING
```

Example:COMmand CONFIG_CHECK

This example checks the configuration, highlighting any pack configuration changes in chosen situations, for example after each Halt Load, every day, etc.

The code at the beginning of this command can be configured for local requirements.

```
%%User configurable variables
$Email:="Operations"; %change to email address which should get error
notices

                                %if run from the schedule.
                                %set to Empty for no email
%%End of user configurable variables
```

If \$Email is assigned a value then an error report will be sent to that address if the command is executed from the schedule and any differences are found from the stored config. If \$Email:=Empty is used and the command is executed from the schedule then a waiting entry will be generated suggesting the use of CONFIG_CHECK from a Supervisor window to check details. The command CONFIG_CHECK can be entered at any time from a Supervisor window (Or TT COM CONFIG_CHECK from an ODT) to report the results of a comparison to the screen.

Once you have modified and saved the file, enter:

```
ENTER COM CONFIG_CHECK FROM OPALS/SUPERVISOR/GETTINGSTARTED
```

Check that the system configuration is as you expect and enter from a Supervisor window:

```
CONFIG_CHECK STORE
```

This will store details of the current system configuration:DLs,SLs,Als and Per=PK. If at any time the system configuration has genuinely changed, use CONFIG_CHECK STORE to store the new config.

Suggested scheduling:

```
TT AF + HL DAILY: TT COM CONFIG_CHECK  
TT AF + 0800 DAILY: TT COM CONFIG_CHECK
```

Example:SITU and ODTs STN_NOTREADY

```
Enter DEF STN_ = FROM OPALS/SUPERVISOR/GETTINGSTARTED
```

These OPALs also detect system messages; this time the messages are output messages from COMS. We are looking for station NOT READY messages and trying to take corrective action by readying the affected stations. The OPALs are invoked as:

```
TT WHEN STN_NOTREADY DO STN_NOTREADY
```

The STN_NOTREADY SITUation checks for a variety of messages by checking the end of their text (using TLIS operator). According to the nature of the message, the string variable \$M is set up with the strings "MON", "RDY" or "CHECK".

In ODTSequence STN_NOTREADY, if the variable M holds the string "MON", then this signifies a station going not ready. The ODTSequence will check that this LSN (isolated from the original message) has not been readied more than 3 times; if not, a READY command is issued via COMS.

If \$M holds the string "RDY" , then the ODTSequence will perform an SM STATUS <lsn> and the displays issued by COMS will be checked by the SITUation.

If \$M holds "CHECK" then the station has still returned not ready after the SM; the ODTSequence will check if the station has been readied more than 3 times before trying yet another READY command.

Others

Other Opal routines may be added to the 'Getting Started' file over time. Each will include a comment describing its use. Some techniques used in these examples can be quite complex. The intent is to make them easy to use. If you have any questions on these techniques or would like some help in using them in your own Opal routines, please contact [Metalogic Support](#).

Copyright

Copyright © 1979-2019 Metalogic S.à R.L., Luxembourg.
All rights reserved.

Last revision: August 2019

METALOGIC believes that the information described in this manual is accurate and reliable, and much care has been taken in its preparation. However, no responsibility, financial or otherwise, is accepted for any consequences arising out of the use of this material.

The information contained herein is subject to change. Revisions may be used to advise of such changes and/or additions.

ACKNOWLEDGEMENT

Many people over a long period of time have contributed to this software.
To all of our contributors, many thanks.

METALOGIC S.à R.L.
PO Box 11
L-7508 Lorentzweiler
Luxembourg

<http://www.metalogic.eu.com>