



# **Metalogic New Feature Paper Supervisor and Unisys NAP Software Interfaces**

**Relative to Supervisor version 42.421.32  
August 29, 1997**

---

## **Introduction**

This paper discusses the availability of interfaces into the Unisys' A-Series Network Application Platform (NAP) offered by Metalogic's SUPERVISOR package; its audience is intended for A-Series sites using NAP who may be using or considering SUPERVISOR to assist with automating day-to-day operations. The special extensions applied to SUPERVISOR, coupled with new facilities in its in-built OPAL language, allows the construction of simple scripts to provide effective monitoring of NAP system events and operations.

For those readers unfamiliar with SUPERVISOR, a brief overview of the its capabilities is covered in the next section. It should be noted that the general content of this paper is somewhat technical in nature and background knowledge of Unisys A-Series and NAP systems would be of benefit. The overall intent of the document is to provide the reader with a concise overview of the interfaces offered by SUPERVISOR into the world of NAP system operations.

## Overview of SUPERVISOR

Metalogic's SUPERVISOR program offers A-Series users a number of benefits by recognising that each and every 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 scripting language for defining operational situations and remedial actions. 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 which 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.

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 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 Object class is defined by a set off 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.
- **SECURITY:** detects an actual or potential security violation.
- **MESSAGE:** allows the detection of all system and application display messages.
- **NAPLOG:** captures log records as they are written away to the NAPLOG system files
- **LOG:** captures all log records as they are written to the system SUMLOG by the MCP.

There are many others, mostly derived from the LOG object class: LOGON, LOGOFF, FILEOPEN, FILECLOSE, TAPELABEL, PRINTS, SILO, COMPLETED etc.

## OPAL and NAP

With reference to the NAP world, SUPERVISOR utilises three different object classes in which OPAL scripts can be written to monitor and sometimes control many aspects of NAP operations.

First, the **SYSTEM** object class includes a subset of attributes, specifically for NAP, which are mapped to information returned by calling entrypoints in the NAP Statistics Library (NSL). These attributes can be used in the construction of simple OPAL scripts to retrieve both static and dynamic NAP-specific information. SUPERVISOR has extensive on-line help facilities to provide concise descriptions of each attribute and a subset of these are shown in Appendix A. Please note that, in the case of the **SYSTEM NAP** attributes, as they are known, those shown here are only a small subset of the list available.

Secondly, the **MESSAGE** object class can be used to detect specific NAP display messages in real-time, using OPAL scripts in a **SITUATION** program to exclude all other irrelevant application messages. The implementation is fast, efficient and is useful for tracking not only NAP events but many other kinds of critical system messages, if required. SUPERVISOR uses a MCP library entrypoint called **REPORT\_LOG\_ENTRIES**, which allows user programs to capture intact log records before they are written by the MCP to the system SUMLOG.

Lastly, in an implementation similar to the **MESSAGE** class, the **NAPLOG** object class uses entrypoints in the NAPSUPPORT library, specifically the **IMPORT\_NAPLOG\_ENTRY** entrypoint, which can capture entire log entries as they are written by the NAP application software to the NAP log files. OPAL provides many attributes which directly map to information held in these log entries, allowing the real-time retrieval of useful diagnostic information which would not normally be available without retrospectively using **NAPLOGANALYZER** to investigate possible problems.

All these implementations are discussed, with examples, in the following sections. In each case, the critical nature of the events being captured and analysed would usually require their reporting to the outside world. In the SUPERVISOR environment, one method of achieving this is to use the Metalogic **RECORDER** program which is passed user-formatted information from OPAL scripts using a special statement called **RECORD**. Upon receipt of such information, the **RECORDER** program can write this data to disk or to a port files; the latter method allows alert software running locally or on remote systems to process the data for further site-specified handling.

## Monitoring NAP operations and environment

The **SYSTEM NAP** subset of attributes is available for use within any OPAL program and can be used to "poll" the NAP system monitoring critical or important NAP system information. As stated earlier, Supervisor uses a fully documented entrypoint in the NAP Statistics Library, called **NSL\_REQUEST**, to extract information into its own set of attributes. Similar to the attributes used in A-Series file and task handling, the OPAL equivalent can be used to describe hardware components, software environmental information and track both NAP and system performance.

The following customer-provided example programs show a method of monitoring the current size of the NAP **VOICEFILE** file by using a **SUPERVISOR SITUATION**. This **SITUATION** employs the OPAL attributes **NAPINUSEVOICESEGMENTS** and **NAPTOTALVOICESEGMENTS** which return, respectively, the number of allocated message segments and the total number of possible message segments in the **VOICEFILE**. The **DEFINE** command is used to "compile" the program into Supervisor's program dictionary.

```
TT DEFINE + SITUATION LOW_VOICE:
  PUT( "VOICE",
      (NAPINUSEVOICESEGMENTS/NAPTOTALVOICESEGMENTS) * 100) GTR 95
```

The accompanying **ODTSEQUENCE**:

```
TT DEFINE + ODTSEQUENCE LOW_VOICE:
  ODT("BEGIN JOB WARNOP; DISPLAY ACCEPT"",
      "WARNING:Voice storage ",STRING8(GET("VOICE"),5),"%";",
      "END JOB");
  RECORD [6]("WARNING: Voice storage at ",GET("VOICE") "%");
```

```
TT WHEN LOW_VOICE DO LOW_VOICE DELAY 15
```

Note: The usage of the **TT** prefix for the above program definitions is required if **SUPERVISOR** commands are used from an **ODT**. From a **SUPERVISOR COMS** window, the use of **TT** is not mandatory since most commands will be accepted without its presence.

The OPAL programs are invoked using a special Supervisor command called **WHEN** which allows a **DELAY** to be specified, here it is 15 minutes. This means that the **LOW\_VOICE SITUATION** will be evaluated at the end of each 15 minute cycle and, if the expression evaluates the in-use component of the **VOICEFILE** as greater than 95%, the associated **ODTSEQUENCE** will be executed.

The **ODT** statement inside the **ODTSEQUENCE** invokes a **WFL** job which gives a waiting entry informing operations that the **VOICEFILE** available space is approaching critical. Immediately following this, the **RECORD** statement will send a message to the **RECORDER** program which forwards it, via port file, to a site-maintained internal alert system.

There are, currently, approximately 100 attributes which can be used to retrieve NAP-related information from the system though this figure is likely to increase as the NAP system software is enhanced.

Some very simple examples of generating information displays from ODTSEQUENCES using the OPAL SHOW statement are shown below. The text inside a SHOW statement is returned directly back to the users terminal. These programs can be defined and used from any ODT or SUPERVISOR COMS session. The use of '/' (slash) at the end of each line signifies a new-line operator.

```
TT DEFINE + ODT$ NAP_AIMSTUFF:
  SHOW( "INCOMING CALLS      = ",AIMINCOMINGCALLREPORTS,/,/,
        "TRANSFERRED DIALOGS = ",AIMTRANSFERREDDIALOGS,/,/,
        "CON.TRANSFED.DIALOGS = ",AIMCONNECTEDTRANSFERREDDIALOGS,/,/,
        "TRANSCATION COUNT   = ",AIMTOTALTRANSACTIONCOUNT,/,/,
        "RESPONSE > 1 SEC.   = ",AIMRESPONSESOVER1SEC,/,/,
        "RESPONSE > 1.5 SEC = ",AIMEXCEEDS1POINT5SECS,/,/,
        "AIMBUCKET[0] <1 SEC = ",AIMBUCKET(0),/,/,
        "AIMBUCKET[1] >1 AND <2 SEC = ",AIMBUCKET(1),/,/,
        "AIMBUCKET[2] >2 AND <3 SEC = ",AIMBUCKET(2),/,/,
        "AIMBUCKET[3] >3 AND <4 SEC = ",AIMBUCKET(3),/,/,
        "AIMBUCKET[4] >4 AND <5 SEC = ",AIMBUCKET(4),/,/,
        "AIMBUCKET[5] >5 AND <6 SEC = ",AIMBUCKET(5),/,/,
        "AIMBUCKET[6] >6 SEC   = ",AIMBUCKET(6));
```

This program displays some of the available information from the NAP system which changes dynamically with time. It can be executed, stand-alone, using the 'DO' command and produces a formatted paged response back to the originating station (ODT or SUPERVISOR COMS window).

```
TT DO NAP_AIMSTUFF

INCOMING CALLS      = 146
TRANSFERRED DIALOGS = 0
CON.TRANSFED.DIALOGS = 0
TRANSACTION COUNT   = 452
RESPONSE > 1 SEC.   = 1
RESPONSE > 1.5 SEC = 82
AIMBUCKET[0] <1 SEC = 451
AIMBUCKET[1] >1 AND <2 SEC = 0
AIMBUCKET[2] >2 AND <3 SEC = 0
AIMBUCKET[3] >3 AND <4 SEC = 0
AIMBUCKET[4] >4 AND <5 SEC = 0
AIMBUCKET[5] >5 AND <6 SEC = 0
AIMBUCKET[6] >6 SEC   = 1
```

Since this information is constantly changing, using OPAL is comparatively easy to monitor changes in critical components like AIMBUCKET data over short periods of time e.g. intervals of 10 seconds or even less.

This second example displays some data about the NAP VOICEFILE and general messaging information:

```
TT DEFINE + ODT5 NAP_MORESTUFF:
  SHOW(
    "NUMBER OF VIODB RECORDS           = ",NAPINUSEVIODDBRECORDS,/,
    "NUMBER OF VOICE MESSAGES          = ",NAPINUSEVOICEMESSAGES,/,
    "NUMBER OF ALLOCATED BUFFERS       = ",NAPALLOCATEDBUFFERS,/,
    "NUMBER OF INUSE BUFFERS           = ",NAPINUSEBUFFERS,/,
    "NUMBER OF ACTIVE VOICE PROCESSES  = ",NAPACTIVEVOICEPROCESSES,/,
    "MAX VOICE BYTES PER BUFFER        = ",NAPVOICEBYTESPERBUFFER,/,
    "NUM TOTAL BYTES PER BUFFER        = ",NAPTOTALBYTESPERBUFFER,/,
    "TOTAL NAP MESSAGES SENT           = ",NAPMESSAGESSENT,/,
    "TOTAL NAP MESSAGES RECEIVED       = ",NAPMESSAGESRECEIVED,/,
    "TOTAL NAP MESSAGES DELETED        = ",NAPMESSAGESDELETED,/,
    "NUMBER OF BUFFER COPIES           = ",NAPBUFFERCOPIES,/,
    "NUMBER OF SEG CACHE HITS          = ",NAPSEGMENTCACHEHITS,/,
    "PHYSICAL I/O READS                = ",NAPPHYSICALIOWREADS,/,
    "PHYSICAL I/O WRITES               = ",NAPPHYSICALIOWWRITES,/,
    "NUMBER OF IN-USE VMMM IOCBS       = ",NAPINUSEIOCBS );
```

Again, the program is executed using the 'DO' command:

```
TT DO NAP_MORESTUFF

NUMBER OF VIODB RECORDS           = 14630
NUMBER OF VOICE MESSAGES          = 13472
NUMBER OF ALLOCATED BUFFERS       = 139
NUMBER OF INUSE BUFFERS           = 106
NUMBER OF ACTIVE VOICE PROCESSES  = 3
MAX VOICE BYTES PER BUFFER        = 32256
NUM TOTAL BYTES PER BUFFER        = 32400
TOTAL NAP MESSAGES SENT           = 276
TOTAL NAP MESSAGES RECEIVED       = 7
TOTAL NAP MESSAGES DELETED        = 3
NUMBER OF BUFFER COPIES           = 0
NUMBER OF SEG CACHE HITS          = 124
PHYSICAL I/O READS                = 10519
PHYSICAL I/O WRITES               = 135
NUMBER OF IN-USE VMMM IOCBS       = 139
```

## Detecting NAP display messages

SUPERVISOR, using OPAL programs that utilise the **MESSAGE** object class, is very efficient and effective at intercepting system and application displays, even when the traffic exceeds thousands of messages per hour. NAP system applications, in particular, can produce significant message traffic making the human detection of important system and NAP events very difficult.

The most important attribute associated with the **MESSAGE** class is **TEXT**; this attribute allows OPAL programs to inspect the content of the message as displayed by the system. OPAL's parsing capabilities allow the text to be easily scanned for specific targets. Within a typical NAP information message, there are a number of fields which identify various software or hardware components and can provide concise information about a NAP system event.

The following code example shows how NAP-specific messages can be detected by scanning the text of the message looking for recognisable patterns.

```
TT DEFINE + SITUATION NAP15ALARM(MESSAGE):
  MSGCAT=MSRDISP AND MSGNO=3 AND          %SIMPLE DISPLAY MESSAGE
  ( (PUT("ALARM",                          %APPLICATION ALARMS
      1000+WHICHOF(TEXT INCL
        {"106/ 42/ 0."          %VMM:unexpected DLP result
         ,"109/ 17/ 0."        %VMM:call not complete
         ,"109/ 112/ 0."       %AIM:Already on hook
         ,"109/ 113/ 0."       %Connect call failed:unknown
         ,"109/ 202/ 0."       %NIUM Execute error
        } ) ) GEQ 1000)

  OR
  ("EVNT: " ISIN TEXT AND
  PUTSTR("SURC",                          %ISOLATE URC
  DECAT(DECAT(DECAT(DECAT(TEXT,"EVNT: ",1),"/",1),"/",1),";",4))
                                          NEQ EMPTY

  AND
  PUT("URC",DECIMAL(GETSTR("SURC"))) > 0 AND
  PUT("ALARM",                              %CRITICAL ALARMS
      2000+ WHICHOF(GET("URC")=
        {0001 %ABORT PROCESSING RESTART DATA SET
         ,0002 %ABORT FILE COMPLEMENT
         ,0007 %LOST TNAUDIT AND VMS ZERO RESTART
         ,0104 %T1 CARRIER ALARM
         ,0106 %T1 REMOTE CARRIER ALARM
         ,0114 %VSM BUFFER MANAGEMENT ERROR
         ,0421 %TIMEOUT WAITING FOR NIU RESPONSE
         ,0427 %NO ANSWER RECEIVED FROM NIU
         ,0488 %Terminate Turbo Array Deallocation
         ,0489 %Start Update NDM
         ,0660 %NIU RESOURCE GROUP EXHAUSTed
         ,0661 %RESOURCE LIMITATION CLEARED
         ,0746 %NO AVAIL PORT IN NIU RG
         ,0861 %NO CO LINKS HAVE BEEN DEFINED
         ,0887 %PROGRAMS WILL NOT DELINK FROM AIM
         ,1202 %PACING Q OVERFLOWED
         ,1251 %SMDI LINK IS BAD : NO MSGS RECEIVED
         ,1264 %SMDI Q OVERFLOWED
```

```

,1703 %PERFORMING ENABLE
,1704 %INCORRECT COMS CONFIGURATION
,1705 %BAD APPLICATION DESIGNATOR
,1740 %NO MSG RCVD . LINK MARKED BAD
,1756 %SMDI LINK INACTIVE
,1757 %NO LINKS ACTIVE FOR NIU
,1762 %DCWRITE TO PRIMARYQ ERROR
,1801 %CALL DISCARDING STARTED
,1812 %NO SPACE FOR VOICE FILE
,1813 %VOICE FILE AT CAPACITY
,1814 %DISK FILE IS RESIDENT BUT NOT AVAILABLE
,1815 %VOICE FILE NEAR CAPACITY
,1816 %NO SPACE TO ALLOCATE DISK AREA
,1831 %NAPSUPPORT OUT OF MEMORY
,1882 %NO MESSAGE NUMBERS AVAILABLE
,1883 %NO DIALOGS AT PROGRAM INITIATION
,2001 %INVALID ELSE CLAUSE
,2002 %INVALID RESULT FROM PROCEDURE
,2004 %PROGRAM ENCOUNTERED A LOGIC ERROR
,2409 %PEP is terminating abnormally .
,2410 %No Indexed Prompt Table found for SPINapp
,2500 %PEP FAULTED
,3503 %QUITTING AT OPERATORS REQUEST
,3512 %USER REQUESTED NDM UPDATE FAILED
,3663 %Map Key association failed
,3664 %Invalid port or group for rule
,3665 %Invalid turbo table rule
,3666 %Mapping association failure
,3667 %Timer is lost
,3668 %Resource group limit reached
}
)) GEQ 2000) )

```

For example, one important subset of NAP events provides display messages which have the following format similar:

```
DISPLAY: EVNT: 45/17/3045;
```

In the above case, the NAP Unique Reason Code (URC) which indicates the nature of the alarm, is the last field in the display (i.e. 3045). The other fields denote the event category and sub-category and all these fields can be retrieved using OPAL string parsing techniques. The OPAL `DECAT` function searches a string for a specific target and, if found, will return a resultant string according to its third parameter. In the above, `DECAT` is used extensively to parse the message text and isolate the URC, using "/" and ";" as targets strings.

Once the URC is known, it is checked against a master list of URCs for which the site wishes to capture and take action. In most cases, this action will probably consist of reporting the event to an external alert system.

However, there are several problems with using a **MESSAGE** class for monitoring NAP systems. First, despite the ability to filter out unwanted entries using an **OPAL SITUATION**, **SUPERVISOR** must apply some processing on all generated messages before passing them to the filter. If the **OPAL SITUATION** is complex, this overhead can be appreciable on systems which have high message traffic.

Secondly, the information displayed by a NAP system message is not, by its nature, very detailed and, although the message can provide a trigger, it is likely that further investigation would be needed to extract more detailed information about the event e.g. by examining the log files using **NAPLOGANALYZER**.

It was primarily this latter reason that led to the implementation of the **NAPLOG** object class, discussed in the next section.

## Detecting NAPLOG events using SUPERVISOR

The **NAPLOG** object class allows the capture of NAP-related events before they are written into the NAP log files. Many of these events have comparable system messages which can be trapped by OPAL programs of the **MESSAGE** class, as discussed in the earlier section, but a raw NAP log record entry contains extremely useful information normally only accessible using the Unisys NAPLOGANALYZER utility. Like **MESSAGE** programs, the **NAPLOG** interface operates in real-time, linking directly to the **NAPSUPPORT** system library, via an entrypoint called **IMPORT\_NAPLOG\_ENTRY**, which is responsible for the forwarding of raw **NAPLOG** information.

Currently, OPAL programs using the **NAPLOG** object are purely event-driven so only the Supervisor **WHEN** command is permitted with **NAPLOG** OPAL programs. However, in later releases, it is intended that the interface will be extended to provide a mechanism to read current and off-line NAP log files, using the **SUPERVISOR EVALUATE** command, in a similar fashion to that used for the recent Supervisor **LOG** object class implementation.

The **NAPLOG** object class offers considerable advantages over the **MESSAGE** variant; the interface is fast, efficient and access to important information such as event severity, unique reason code and component data is available by mapping them directly to OPAL attributes.

Event severity codes currently range from 1 (Normal) to 10 (NAP system inoperable), increasing in importance. Each event has its own unique reason code (URC) or message number which can be used to identify an individual NAP event which are described, in detail, in an Unisys provided file called **NAPSYMBOL/LOG/MESSAGES**. So, for example, **NAPLOG** attributes exist such as **URC** and **SEVERITY** will return the event reason code and its severity level respectively for the NAP log entry being processed.

A special OPAL attribute, **LOGTEXT**, provides a text representation of the analysed message, similar to that produced by the Unisys **NAPLOGANALYZER** utility. Indeed, other **NAPLOG** attributes, such as **REASONTEXT** and **ADDLPARAM**, are assigned to various text elements in the **LOGTEXT** string which avoid unnecessary string parsing by OPAL.

Some example OPAL programs are shown below:

```
TT DEFINE + SITU NAPMONITOR(NAPLOG):  
    SEVERITY GTR 3 AND URC NEQ {4000,4001,5023,6000}
```

```
TT DEFINE + ODT'S NAPMONITOR(NAPLOG);  
    RECORD[ 6 ] (TIMESTAMP, , "SEVERITY=", SEVERITY, , " ,CODE=", EVENTREASON);  
    RECORD[ 6 ] ( " " ,REASONTEXT);
```

These programs are initiated using the **SUPERVISOR WHEN** command:

```
TT WHEN NAPMONITOR DO NAPMONITOR
```

This SITUATION will filter all NAPLOG object class events with severity 3 or higher but will exclude reason codes 4000,4001,5023 and 6000. This is because even though their severity classification is high, it may be required to exclude certain known entries that do not need tracking. In this case, the message numbers have been chosen at random, for example only, and are not meant to reflect the requirements of a real NAP environment. As with previous examples, the associated ODTSEQUENCE is performing RECORD statements, so these messages are being sent to an alert system or journal printer.

For example, a typical event alert might appear as :

```
11:38:04 11/12/96 SEVERITY=6, CODE=804
REASON 804: NETWORK TERMINATED A CALL SETUP
```

A more detailed example SITUATION is shown below, with thanks to Charlie Uhlman at Pacific Telesys:

```
TT DEF + SITU NAPM (NAPLOG):
  URC = {104          % T-1 carrier alarm
        ,148          % VIM buffer underrun
        ,404          % % Too many SMDI's without seizures
        ,427          % Hung front end port
        ,664          % NIU card offline
        ,758          % Hung back end port
        ,759          % Hung back end port
        ,945          % TSP board failure
        ,964          % TSP initialization failure
        ,1803         % Throttle info...discarding continues
        ,1812         % No space for the voicefile
        ,1813         % Voicefile at capacity
        ,1853         % Resource group WARNING
        ,1855         % Resource group EXHAUSTED
        ,1856         % Resource group EXHAUSTED for NIU
        ,3421         % VNMS terminating calls for TSP
        ,5610         % Fatal PDP failure
        ,5613         % TSP initialization failure
        ,5628         % TSP board failed heartbeat (FAILED)
        ,5629         % TSP board failure
        ,5913         % SS7 link out of service at level 2
        ,5915         % SS7 link out of service SLT failure
        ,5918         % Fatal (invalid system call)
        ,5920         % No response from PRIM for SS7 link
        ,5924         % SS7 link failure
        }
OR                                     % Get everything at severity 8
( SEVERITY GEQ 8 )
```

This code constitutes a filter which uses the URC attribute to check it against a 'master' list of eligible errors codes but any event which has a severity level greater than 8 will always be automatically captured by the SITUATION code.

LOGTEXT is probably one of the most useful attributes available but should not be used in a SITUATION as there is CPU overhead associated with its creation. Although LOGTEXT is optimised such that, once accessed, the generated text is cached internally and retained for the life of the log entry, it is recommended that its usage is kept to a minimum and only used to provide additional information once a NAP log event is being processed.

The example SITUATIONS shown earlier both use the SEVERITY and URC attributes to filter unwanted error codes and the LOGTEXT attribute is only used in the associated ODTSEQUENCE to extract additional information. Note that use of the attributes REASONTEXT, ADDLPARAM and LINE will automatically generate the LOGTEXT attribute when necessary.

Although the ODTSEQUENCES shown here are very simple, little effort would be needed to transform the an event into a meaningful English message, translating component information into geographical locations, telephone numbers etc. For example, consider the COMPONENTINFO and COMPONENT attributes which provide all or part of the component details of a NAP event usually seen in the display message. For event error codes 1231, 1232 and 1256, which correspond to SMDI link problems, COMPONENT(3) identifies the failing link station (LSN) involved in the event.

So, instead of generating an alert message of the form:

```
SMDI LINK FAILURE FOR LSN 41
```

it might be desirable to replace it with a more informative message:

```
SMDI LINK FAILURE FOR GREENLAND: (555)-916 -5555
```

One simple OPAL script can be used to hold the rules that controls the translation of logical station number 41, as returned by the COMPONENT(3) attribute, into the above location and telephone number. Such a database of rules is easily maintained in OPAL scripts held in the SUPERVISOR program dictionary.

The full attribute subset for the NAPLOG object is shown in Appendix A.

## Implementation of the NAPLOG interface

Much of the functionality for the implementation of the `NAPLOG` object class in `SUPERVISOR` is provided by a Metalogic supplied freeze-controlled library, whose function name is called `METANAPLIB`. The released codefile title is called:

`*METALOGIC/SUPERVISOR/NAPINTERFACE`

This library currently supports NAP releases from 16 to 17.3 and provides most of the intelligence needed to understand and handle NAP log events. `SUPERVISOR` itself does not link to the `METANAPLIB` library but uses a subordinate task, called `NAPHANDLER`, to perform the linkage process and calls `METANAPLIB` entrypoints. Due to the critical importance of `SUPERVISOR` presence in the system, this choice reduces the possibility that it might be affected by NAP failures or outages.

The `METANAPLIB` library is also responsible for linking to two NAP system libraries, `NAPSUPPORT` and `NSL`, which have various entrypoints that support both the `NAPLOG` and `SYSTEM NAP` object implementations.

### **METANAPLIB operation and installation**

If the `METANAPLIB` has been installed before, the Metalogic `INSTALL` utility will automatically load and `SL` the library during normal installation. However, the first installation will require the software to be set up manually with the following `ODT` commands:

```
COPY *METALOGIC/SUPERVISOR/NAPINTERFACE FROM META4242021
```

```
SL METANAPLIB = *METALOGIC/SUPERVISOR/NAPINTERFACE
```

where '`META4242021`' is the name of the Metalogic release tape. If necessary, the symbolic and a compilation job are available on the release tape if the library requires changes or re-compilation. These files are shown below:

```
(METASOFT) SUPERVISOR/NAPINTERFACE
```

```
(METASOFT) NAPCOMPILE/=
```

Please note that the library codefile has `PU` privilege and is assigned an `IDENTITY` of "`NAPINT`" making it easier to detect `METANAPLIB` related display messages.

Two new Supervisor **TT USE** commands have been implemented to assist with the set-up of configuration information required by the METANAPLIB library:

**TT USE FILE <file name> FOR NAPRUNLIGHT**

**TT USE TASK <codefile name> FOR NAPSTATSLIB**

In the above, the NAPRUNLIGHT file name is used to allow METANAPLIB to check whether the NAP system is active since NAP will exclusively lock this file during its initialization. Usually, the NAPRUNLIGHT file is called:

**\*NAP/RUNNING/LIGHT ON <FAMILYNAME>**

Secondly, the NAPSTATSLIB library name is used by the METANAPLIB library to determine the name and location of the NAP Statistics Library (NSL), which is typically called:

**\*NAPSYSTEM/NSL ON <FAMILYNAME>**

In both cases, full titles are necessary including each files normal family name.

In addition, the METANAPLIB library also checks for the presence of a second file :

**\*NAPSYSTEM/LIVEFILE**

which is expected to be found on the same family as the NAPRUNLIGHT file. The METANAPLIB library uses the presence of this file as a secondary confirmation that NAPSUPPORT is initialised and active. Although the NAPRUNLIGHT file may already be open exclusive by NAPSUPPORT, it may be that the library has been delayed during its termination or initialisation phases.

This LIVEFILE check ensures that METANAPLIB does not prematurely attempt to connect to NAPSUPPORT whilst the library may be in an unknown state, especially if the library is terminating.

## Using METANAPLIB

SUPERVISOR will only attempt to connect to the METANAPLIB library whenever an OPAL program accesses a **SYSTEM NAP** attribute or **NAPLOG** object programs have been initiated with the **WHEN** command. SUPERVISOR will invoke the **NAPHANDLER** sub-task to handle the initial library linkage to METANAPLIB. The **NAPHANDLER** process will then remain active, handling all NAP-related event and information requests.

During METANAPLIB initialisation, various checks are made to ensure that the SUPERVISOR-NAP configuration is acceptable; any failures will cause the automatic abort of the METANAPLIB library and the requesting OPAL programs will be forced into a waiting condition.

If the **NAPRUNLIGHT** configuration item (set using **TT USE**) is incorrect then METANAPLIB will abort with the message:

```
NAP_RUNLIGHT CONFIG UNKNOWN OR ILLEGAL
```

If the **NAPSTATSLIB** configuration item (also set using **TT USE**) is incorrect then METANAPLIB will abort with the message:

```
NAP_STATSLIB CONFIG UNKNOWN OR ILLEGAL
```

If METANAPLIB is unable to link to the **NSL** library, probably due to an incorrect library title being used for **NAPSTATSLIB**, it will abort with

```
UNABLE TO LINK TO NAP STATISTICS LIBRARY:<n>
```

where **<n>** signifies the error value returned by the **DCALGOL LINKLIBRARY** function.

If METANAPLIB fails to get the **NAPSUPPORT** library name using the **NAP\_CONFIG** entriypoint because of an internal error, it will abort with following message:

```
ERROR FROM NAP_CONFIG INTERFACE
```

In the above cases, the METANAPLIB library will abnormally terminate and the **NAPHANDLER** will be informed of the forced de-linkage. SUPERVISOR will display the message:

```
SUPERVISOR:DELINKED FROM METANAPLIB LIBRARY
```

In these circumstances, while there is a NAP requesting OPAL program, the **NAPHANDLER** will wait 45 seconds before re-trying a connection to a new invocation of the METANAPLIB library.

Next, if either the **LIVEFILE** or **NAPRUNLIGHT** files are not both in-use, then the METANAPLIB library will issue an **ACCEPT** message to advise the operator :

```
ACCEPT: METANAPLIB: WAITING FOR NAP SYSTEM AVAILABILITY
```

From this point, METANAPLIB will check every 30 seconds until the files are in the correct state; METANAPLIB will then automatically acknowledge the **ACCEPT** and link to both the **NAPSUPPORT** and **NSL** libraries before becoming a freeze-controlled library.

Whilst METANAPLIB is unable to successfully complete initialisation because of any of the previous reasons, the status of any SUPERVISOR initiated NAPLOG or SYSTEM NAP OPAL programs will be marked as (WAIT NAP) when interrogated from any ODT or SUPERVISOR window:

```
TT WHEN ? NAP=

----- SUPERVISOR WHEN STATUS (Limit = 40, Active = 1, Queued = 0) -----
W 000 7547 WHEN NAPLOG_MONITOR DO  NAPLOG_MONITOR      (WAIT NAP)
                                     TIMES:CPU=00:00:08,IO=00:00:00,ET=1:27:04
      550 evals, 35 ODTs entries, 1 Restarts
```

During initialisation, METANAPLIB will use an entrypoint in NSL, called NAP\_CONFIG, to determine the name of the NAPSUPPORT library. If this linkage fails, then SUPERVISOR will abort any waiting OPAL programs with the message:

```
SUPERVISOR: BAD LINKAGE TO NAPSUPPORT LIBRARY
```

The most likely reason for this error is an incompatibility problem with the declared library entrypoints in METANAPLIB compared with those exported from the NAP Support library. If this occurs, Metalogic should be contacted as soon as possible for advice.

If the METANAPLIB library has successfully initialised, the interface between SUPERVISOR and the NAP library software will now be active. Any NAPLOG programs will begin processing NAP log events and SYSTEM NAP requests will be processed.

Once a NAPLOG slot is active, the METANAPLIB library control process will be linked into the IMPORT\_NAPLOG\_ENTRY entrypoint in NAPSUPPORT and begin processing events. Because of the differences in the implementation of this interface by Unisys between NAP releases 16 and 17, the METANAPLIB control process remains active and linked to the IMPORT\_NAPLOG\_ENTRY entrypoint even when the last NAPLOG program has terminated. Note that ways of terminating METANAPLIB are discussed in the section on "Handling NAP terminations" later in this document.

In normal circumstances, there should only ever be one user of the METANAPLIB library and that should be Metalogic's Supervisor program.

### **Note for users on NAP releases earlier than 17.3**

On NAP 17.3 and later, the METANAPLIB library will gracefully terminate from the IMPORT\_NAPLOG\_ENTRY. However, on earlier releases of NAP, the termination of the METANAPLIB library will occupy a "registration" slot in the NAPSUPPORT library once a NAPLOG WHEN has been activated. It is understood that only 20 such slots are available during normal NAP running and it follows that only 20 invocations of the METANAPLIB will be permitted. On pre-NAP 17.3 systems, only a full NAP system restart will reallocate the full 20 slots.

The METANAPLIB library can be DS-ed, if absolutely necessary, and will force any active NAP WHENs to automatically return to a (WAIT NAP) state. However, this procedure is NOT recommended except in extreme circumstances since a forced DS will use up a registration slot even on later NAP releases.

### **De-implementation of the OPALNSLLIB library**

As already discussed earlier, Supervisor supports a set of non-event NAP attributes in the `SYSTEM` object class. These attributes retrieve information from the NAP Statistics library and, on earlier releases, Supervisor used an intermediary SL-ed library, called OPALNSLLIB, which performed the linkage and information retrieval from the statistics library. Because of the implementation of METANAPLIB, OPALNSLLIB functionality has now been merged into METANAPLIB and the old library has been deimplemented.

Once the new METANAPLIB library and SUPERVISOR have been installed, customers still using OPALNSLLIB should remove the old library:

```
SL - OPALNSLLIB
```

All OPAL programs that use these static NAP attributes do not require re-compilation since the original OPALNSLLIB calls will be automatically re-directed to the METANAPLIB library instead.

## Handling NAP terminations

If the NAPSUPPORT is forcibly terminated whilst SUPERVISOR NAPLOG programs are active, the METANAPLIB control process will be automatically terminated by the MCP as a user of that library. During its termination process, METANAPLIB uses an epilog procedure to request SUPERVISOR to manually delink from the library, thus avoiding a similar affect on SUPERVISOR when METANAPLIB has terminated.

In such circumstances, the NAPHANDLER process will halt for 45 seconds and then try to restart the METANAPLIB library. Since NAPSUPPORT can often take a long time to terminate, this ensures that the NAPHANDLER process does not immediately attempt to link to what might seem to be a valid NAPSUPPORT library.

METANAPLIB will also re-check that the LIVEFILE and NAPRUNLIGHT files are in the correct in-use state before attempting to re-link to NAPSUPPORT. If they are not, METANAPLIB will use the same ACCEPT mechanism discussed earlier.

Any OPAL program requesting access to a SYSTEM NAP attribute or active NAPLOG programs will be immediately marked with a status of (WAIT NAP) when interrogated by a TT WHEN ? command.

Should it be necessary to disable SUPERVISOR's NAP interface totally, all NAP-related programs should be terminated followed by the METANAPLIB library. This can be achieved by the following system commands:

```
TT DELINK METANAPLIB
```

```
<mixno METANAPLIB>AX QUIT          or  
<mixno METANAPLIB>THAW
```

Using the DELINK command is only sufficient to force the METANAPLIB library to have its library connection broken with the NAPHANDLER task. Unless the SL function has been deleted (SL-) or reassigned to a new codefile version, the library will remain active until the QUIT or THAW is processed.

## Error code (URC) filtering

Although the handling of individual NAP log entries is very efficient, SUPERVISOR has the ability to advise the NAPSUPPORT library to filter out certain ranges of error codes from being processed by METANAPLIB in the first place. This mechanism is achieved by passing an array mask to another entrypoint in the NAPSUPPORT library, called ALTER\_NAPLOG\_FILTER, which alters the range of allowable URC numbers returned to METANAPLIB by the IMPORT\_NAPLOG\_ENTRY procedure. This feature has a significant efficiency benefit in that SUPERVISOR will be automatically screened from processing specific URC events that will never be required by a NAPLOG program.

The NAPLOG object class will accept a numeric subcontext in a SITUATION definition:

```
TT DEFINE + SITUATION NAP_MONITOR(NAPLOG=3) :
```

In this case, SUPERVISOR will search for a configuration variable named NAPFILTER\_3 which has already been set up using the Metalogic INSTALL utility from CANDE:

```
U META/INSTALL NAPFILTER_3=3721,4567,1000-1040,890
```

Each number or range in the list should be separated from its neighbours by a comma or space. Multiple filters can be used simply by changing the number suffix. When a NAPLOG program with a filter assignment is invoked, the NAPHANDLER task will display an information message and check the validity of the filter values. If any errors are detected, an appropriate error is displayed and the filter will be discarded.

If the check on the filter range is successful, then NAPHANDLER will call ALTER\_NAPLOG\_FILTER, passing the filter values as a parameter. This will inhibit the forwarding of any NAP log entries whose URC code is marked in the filter.

This mechanism offers an ideal way of excluding common and unwanted NAP log entries, known by their URC, which would otherwise consume SUPERVISOR resource. Although SUPERVISOR will allow multiple NAPLOG programs to use filters, the most efficient environment is a single NAPLOG program with its own extensive filter.

## Alert systems and SUPERVISOR

Although SUPERVISOR makes the best use of standard A Series resources for reporting (messages to ODTs or COMS windows, 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 implemented the HOTLINE and RECORDER mechanisms.

Within an OPAL program, it is possible, by use of the RECORD verb, to direct user formatted output to a specific disk file. In addition, this information can optionally be directed to a BNA Port File interface. This means, for example, that critical system events can be detected and alert messages directed through BNA 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; such as COMENSA and Command-Post; others have linked into paging systems so 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 tape. The OPAL verb RECORD passes a string from an ODTSEQUENCE to the RECORDER program, generally for the purpose of logging some condition to a disk file or passing information to HOTLINE. RECORDER and HOTLINE are designed to be user-customisable.

Additionally, the RECORD statement requires a file number, hereafter known as "message class", which controls whether the message is written to disk or port file. If the message class is from 0 to 5, the message is always written to disk.

The generated RECORDER disk files are named:

```
*SUPERVISOR/RECORD/<dayname>/<filename>.
```

Where <dayname> is the day of the week that the file was created (e.g. MONDAY, TUESDAY, etc.). The files are placed on the system DL BACKUP family. The files are created as standard JOBSYMBOL files but have PROTECTION=PROTECTED to limit record loss due to a system failure.

The same OPAL syntax is used as for logging to a PORT file except that the message class ranges from 6 to 47. RECORDER will send data to any HOTLINE program which has requested that class of message.

The supplied HOTLINE program can be run from any remote station. The program will ask the operator what hosts, and what message classes, to monitor. The operator can select message classes 6 through 9 singly, or can select all message classes (6 through 47). 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.

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 it is necessary to identify the specific source OPAL, the RECORD statement can be easily changed to include this 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.

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 through the ?ON <window> command and a terminal can be dedicated to this environment.

Also, some sites have applied simple changes to HOTLINE to provide a TCP/IP interface which can permit the routing of SUPERVISOR messages directly to a specified UNIX or PC, running its own alert software, simply by assigning an IP address. This approach provides even greater flexibility if TCP/IP is available on the A-Series network.

## Summary

SUPERVISOR offers many facilities to a non-NAP system including the real-time detection of messages, waiting entries, job and tape handling and tracking of all events that have been logged to the system SUMLOG. In practice, most system tasks that can be performed by an operator can be handled by SUPERVISOR.

In the NAP environment, the availability of the NAPLOG object class allows the real-time detection of important hardware or software problems by trapping events before they are written to the NAP log. The flexibility and simplicity of the OPAL language means that powerful scripts can be written to provide message filtering and an engine to drive port file or TCP/IP interfaces to a site supported alert mechanism.

Further, SUPERVISOR's message detection capabilities can be used to track any specific NAP application displays that do not have corresponding NAP log entries and, also, other critical system messages. These might include, in the latter case, examples such as the loss of the last path to a critical unit or RF degradation messages.

Lastly, SUPERVISOR has many facilities for monitoring and extracting data from the NAP environment, allowing rapid and concise ad-hoc reporting or to poll various critical resources, such as the VOICEFILE, over time.

## Appendix A: OPAL SYSTEM NAP attribute subset

The following list of attributes is readily available from SUPERVISOR using the **TT PRINT ATT** command. Only a small subset of the more interesting NAP-specific attributes are shown here:

- AIMBUCKET** Returns INTEGER  
Parameters : 1. Integer range 0 to 6  
Semantics : A count of command/response pairs since NAP initialisation grouped into durations of < 1 second, between 1 & 2 seconds, 2 & 3 seconds, 3 & 4 seconds, 4 & 5 seconds, 5 & 6 seconds and 6 or more seconds.
- AIMCALLCONNECTEDRESPONSES** Returns INTEGER  
Semantics : The total number of Call Connected Responses received by AIM since NAP initialisation.
- AIMTOTALTRANSACTIONCOUNT** Returns INTEGER  
Semantics : The total number of response/command pairs processed since NAP initialisation.
- AIMTRANSFERREDDIALOGS** Returns INTEGER  
Semantics : The total number of incoming calls that have been transferred.
- NAPALLOCATEDBUFFERS** Returns INTEGER  
Semantics : The number of digital voice data buffers currently allocated by the VMMM.
- NAPBUFFERLIMIT** Returns INTEGER  
Parameters : 1. Integer range 1 to 3  
Semantics : A buffer control parameter that specifies a non-binding limit on the number of buffers in this size group that can be allocated by NAP.
- NAPINUSEBUFFERS** Returns INTEGER  
Semantics : The number of digital voice data buffers currently being used by the VMMM to transfer digital voice data.
- NAPINUSEIOCBS** Returns INTEGER  
Semantics : The number of in use vmmm I/O control blocks.
- NAPINUSEVIODBRECORDS** Returns INTEGER  
Semantics : Number of records in the VIODB data base that describe allocated voice messages.
- NAPINUSEVOICEMESSAGES** Returns INTEGER  
Semantics : The number of allocated voice messages.
- NAPINUSEVOICESEGMENTS** Returns INTEGER  
Semantics : The number of voice message segments in the VOICEFILE files that are components of allocated voice messages.
- NAPTOTALVOICESEGMENTS** Returns INTEGER  
Semantics : The number of voice message segments in the VOICEFILE files.

TSPBOARDCOMPONENTS      Returns INTEGER  
Parameters : 1. Integer  
Semantics : Number of TSP board components.

TSPBOARDSLOTNUMBER      Returns INTEGER range 0 to 255  
Parameters : 1. Integer  
              2. Integer range 0 to 9  
Semantics : TSP board slot number

TSPBOARDSTATUS          Returns INTEGER mnemonic  
Mnemonic values : BDCAMPMAINT , BDONLINESIGNAL , BDONLINE ,  
BDNOTPRESENT , BDINITIALIZED , BDSYNCHRONIZED .  
Parameters : 1. Integer  
              2. Integer range 0 to 9  
Semantics : TSP board status

TSPBOARDTYPE            Returns INTEGER mnemonic  
Mnemonic values : PDPBOARD , PDSPBOARD , HIPBOARD , PRIMBOARD .  
Parameters : 1. Integer  
              2. Integer range 0 to 9  
Semantics : TSP board type

## Appendix B: OPAL MESSAGE attribute subset

This complete subset of attributes may be used by OPAL programs of the MESSAGE object class to help identify and analyse application and system display messages.

**IDENTITY** Returns STRING

MSGIDENTITY is a synonym for this attribute

Semantics : MSGIDENTITY returns the IDENTITY of the task that issued the message, as established by the MP <file> + IDENTITY command.

**JOBNO** Returns REFERENCE TO MX

SESSIONNUMBER is a synonym for this attribute

SESSIONNO is a synonym for this attribute

JOBNUMBER is a synonym for this attribute

Semantics : JOBNUMBER returns the job number or session number of the task which displayed the message. Some messages do not have an associated job or session.

**JOBNUMBER** Returns REFERENCE TO MX

SESSIONNUMBER is a synonym for this attribute

SESSIONNO is a synonym for this attribute

JOBNO is a synonym for this attribute

Semantics : JOBNUMBER returns the job number or session number of the task which displayed the message. Some messages do not have an associated job or session.

**MIXNO** Returns REFERENCE TO MX

MIXNUMBER is a synonym for this attribute

Semantics : MIXNUMBER returns the mix number of the task which displayed the message. Some messages do not have an associated task.

**MIXNUMBER** Returns REFERENCE TO MX

MIXNO is a synonym for this attribute

Semantics : MIXNUMBER returns the mix number of the task which displayed the message. Some messages do not have an associated task.

**MSGCAT** Returns INTEGER mnemonic

Mnemonic values : MSRX , MSRHI , MSRDRC , MSRLM , MSRCP , MSRIC , MSRFA , MSREH , MSRHL , MSRAR , MSRDRC , MSRTCP , MSRFAM , MSRLIB , MSRRSF , MSROMR , MSRRES , MSRCTP , MSRPRQ , MSRMIR , MSRDRC , MSRCMP , MSRFOUT , MSRSORT , MSRFORT , MSRSOFT , MSRLIST , MSRWARN , MSRSECV , MSRCSCP , MSRPORT , MSRWARN , MSRTOCNT , MSRLOGIO , MSRPREFMT , MSRRESIZE , MSRATTERR , MSRNETMSG , MSRPATHRES , MSRPHYOERR , MSRNETEXMSG , MSRCONFIGERR , MSRUNITMOVER , MSRDLPRECONFIG , MSRIO , MSRCC , MSRDLC , MSRRS , MSRVL , MSRVS , MSRKF , MSRFR , MSRRP , MSRFIN , MSRIPC , MSRCAT , MSRCAN , MSRFIB , MSRMEM , MSRDPC , MSRGSS , MSRHDR , MSRCON , MSRACC , MSRRTZ , MSRMATH , MSRDISP , MSRINTR , MSRSWAP , MSRMISC , MSRCTRL , MSRCERR , MSRDUMP , MSRSTDIO , MSREBDMS , MSRDBATT , MSRDCERR , MSRCONFIG , MSRPRINTS , MSRSECMMSG , MSRBNAMMSG , MSRMLSDISP , MSRDLPCLR , MSRLOADHOST , MSRSTARTUNIT , MSRSTATUSERR .

Semantics : MSGCAT gives the major MSG type by mnemonic. The mnemonic is the same as the header to the message in a LOGANALYZER output (without the MSGNO appended.) Please note that MSRBNAMSG and MSRNEXMSG categories may only be detected on MCPs with an optional Metalogic supplied MCP patch.

MSGDATE Returns STRING as "DD/MM/YY"  
Semantics : MSGDATE returns the date the message was displayed in Military date format.

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

MSGNO Returns INTEGER range 0 to 4095  
MSGNUMBER is a synonym for this attribute  
Semantics : MSGNO gives the MSG number. The MCP uses this number to find the message text for the given MSGTYPE.

MSGNUMBER Returns INTEGER range 0 to 4095  
MSGNO is a synonym for this attribute  
Semantics : MSGNO gives the MSG number. The MCP uses this number to find the message text for the given MSGTYPE.

MSGPARAM Returns STRING  
Semantics : MSGPARAM returns the text of the message parameter.

MSGTIME Returns REAL in seconds  
MSGTIMESTAMP is a synonym for this attribute  
Semantics : gives the time the message was logged in the SUMLOG by the MCP, in seconds.

MSGTIMESTAMP Returns REAL in seconds  
MSGTIME is a synonym for this attribute  
Semantics : gives the time the message was logged in the SUMLOG by the MCP, in seconds.

MSGTYPE Returns INTEGER range 0 to 4095  
Semantics : MSGTYPE gives the major MSG type number. The MCP has about 70 different message types.

SESSIONNO Returns REFERENCE TO MX  
SESSIONNUMBER is a synonym for this attribute  
JOBNUMBER is a synonym for this attribute  
JOBNO is a synonym for this attribute  
Semantics : JOBNUMBER returns the job number or session number of the task which displayed the message. Some messages do not have an associated job or session.

SESSIONNUMBER Returns REFERENCE TO MX  
SESSIONNO is a synonym for this attribute  
JOBNUMBER is a synonym for this attribute  
JOBNO is a synonym for this attribute

Semantics : JOBNUMBER returns the job number or session number of the task which displayed the message. Some messages do not have an associated job or session.

TEXT Returns STRING  
Semantics : TEXT returns the text of the message received.

## Appendix C: OPAL NAPLOG attribute subset

This is a comprehensive attribute subset that maps directly onto data fields held in individual NAPLOG entries. LOGTEXT is especially useful in that it can reproduce the NAPLOG record as if it had been analysed by NAPLOGANALYZER. Although SUPERVISOR optimises its usage, it is expensive to construct and should be used with some care.

**ADDLPARAM** Returns STRING

ADDLPARAMETER is a synonym for this attribute

Parameters : 1. Integer range 0 to 255

Semantics : Returns the specified additional parameter as a text string.

**ADDLPARAMETER** Returns STRING

ADDLPARAM is a synonym for this attribute

Parameters : 1. Integer range 0 to 255

Semantics : Returns the specified additional parameter as a text string.

**CALLID** Returns INTEGER

Semantics : The Call ID associated with the current dialog, if present.

**COMP\_NIUCARDEPORT** Returns INTEGER range 0 to 4095

Semantics : For COMPONENTCATEGORY=NIUVIMCARD, the NIU card ending port number of this component.

**COMP\_NIUCARDLEVEL** Returns INTEGER range 0 to 7

Semantics : For COMPONENTCATEGORY=NIUVIMCARD, the NIU card rack level of this component.

**COMP\_NIUCARDRACK** Returns INTEGER range 0 to 3

Semantics : For COMPONENTCATEGORY=NIUVIMCARD, the NIU card rack number of this component.

**COMP\_NIUCARDSLOT** Returns INTEGER range 0 to 31

Semantics : For COMPONENTCATEGORY=NIUVIMCARD, the NIU card slot number of this component.

**COMP\_NIUCARDSPAN** Returns INTEGER range 0 to 31

Semantics : For COMPONENTCATEGORY=NIUVIMCARD, the NIU card span of this component.

**COMP\_NIUCARDSPORT** Returns INTEGER range 0 to 4095

Semantics : For COMPONENTCATEGORY=NIUVIMCARD, the NIU card starting port number of this component.

**COMP\_TSPHANDLE** Returns INTEGER range 0 to 255

COMP\_TSPRFSHANDLE is a synonym for this attribute

Semantics : For COMPONENTCATEGORY=TSPFILESERVER, the TSP handle of the component.

**COMP\_TSPRFSBYTES** Returns INTEGER range 0 to 65535  
 Semantics : For COMPONENTCATEGORY=TSPREMOTFILE, the number of bytes involved in this TSP event.

**COMP\_TSPRFSHANDLE** Returns INTEGER range 0 to 255  
 COMP\_TSPHANDLE is a synonym for this attribute  
 Semantics : For COMPONENTCATEGORY=TSPREMOTEFIE, the handle number of the TSP involved in this event.

**COMP\_TSPSCSCICOMMAND** Returns INTEGER range 0 to 255  
 COMP\_TSPTRACENIUUPRIM is a synonym for this attribute  
 COMP\_TSPTIMENIUUPRIM is a synonym for this attribute  
 COMP\_TSPSIGLINKSET is a synonym for this attribute  
 Semantics : For COMPONENTCATEGORY=TSPSCSI, the command processed by the TSP involved in this event.

**COMP\_TSPSCSCIERROR** Returns INTEGER range 0 to 255  
 COMP\_TSPSIGLINKNUMBER is a synonym for this attribute  
 COMP\_TSPTIMENIUCHAN is a synonym for this attribute  
 Semantics : For COMPONENTCATEGORY=TSPSCSI, the error code of the command issued by the TSP in this event.

**COMP\_TSPSIGLINKNUMBER** Returns INTEGER range 0 to 255  
 COMP\_TSPTIMENIUCHAN is a synonym for this attribute  
 COMP\_TSPSCSCIERROR is a synonym for this attribute  
 Semantics : For COMPONENTCATEGORY=TSPSIGNALING, the LINK number of this component.

**COMP\_TSPSIGLINKSET** Returns INTEGER range 0 to 255  
 COMP\_TSPSCSCICOMMAND is a synonym for this attribute  
 COMP\_TSPTRACENIUUPRIM is a synonym for this attribute  
 COMP\_TSPTIMENIUUPRIM is a synonym for this attribute  
 Semantics : For COMPONENTCATEGORY=TSPSIGNALING, the LINK set number of this component.

**COMP\_TSPSLOT** Returns INTEGER range 0 to 255  
 COMP\_TSPTRACELINEEVENT is a synonym for this attribute  
 Semantics : For COMPONENTCATEGORY=TSPFILESERVER, the TSP slot id of the component.

**COMP\_TSPTIMENIUCHAN** Returns INTEGER range 0 to 255  
 COMP\_TSPSIGLINKNUMBER is a synonym for this attribute  
 COMP\_TSPSCSCIERROR is a synonym for this attribute  
 Semantics : For COMPONENTCATEGORY=TSPTIMESLOT, the NIU channel number of this component.

**COMP\_TSPTIMENIUUPRIM** Returns INTEGER range 0 to 255  
 COMP\_TSPSCSCICOMMAND is a synonym for this attribute  
 COMP\_TSPTRACENIUUPRIM is a synonym for this attribute  
 COMP\_TSPSIGLINKSET is a synonym for this attribute  
 Semantics : For COMPONENTCATEGORY=TSPTIMESLOT, the NIU prim number of this component.

**COMP\_TSPTRACELINEEVENT** Returns INTEGER range 0 to 255  
 COMP\_TSPSLOT is a synonym for this attribute

Semantics : For COMPONENTCATEGORY=TSPLINETRACE, the line event of the TSP involved in this event.

COMP\_TSPTRACELINESTATE Returns INTEGER range 0 to 255  
Semantics : For COMPONENTCATEGORY=TSPLINETRACE, the line state of the TSP involved in this event.

COMP\_TSPTRACELINESTATUS Returns INTEGER range 0 to 65535  
Semantics : For COMPONENTCATEGORY=TSPLINETRACE, the line status of the TSP involved in this event.

COMP\_TSPTRACENIUUPRIM Returns INTEGER range 0 to 255  
COMP\_TSPSCSCICOMMAND is a synonym for this attribute  
COMP\_TSPTIMENIUUPRIM is a synonym for this attribute  
COMP\_TSPSIGLINKSET is a synonym for this attribute  
Semantics : For COMPONENTCATEGORY=TSPLINETRACE, the NIU prim number involved in this event.

COMPONENT Returns STRING  
Parameters : 1. Integer range 0 to 255  
Semantics : Returns the requested entry from the COMPONENTINFO attribute.

COMPONENTCATEGORY Returns INTEGER mnemonic  
Mnemonic values : NIURG , MODULE , NAPFILE , NIUCARD , NIUCOMP , SMDIPOINT , VOICEUNIT , APPLERROR , NIULINKLSN , NIUVIMCARD , NAPDATABASE , TSPTIMESLOT , VOICECHANNEL , TSPLINETRACE , CENTRALOFFICE , S4SYSTEMALARM , TABLE , TSPDPC , NIUUNIT , TSPCOMP , TSPSCSI , SMDILINK , TSPSLOTID , TSPLINKSET , TSPPROCESS , CODIRECTORY , TSPOLDTRACE , TSPSIGNALING , TSPREMOTEFIELD , S4NOHOSTALARM , NIUCONTROLINK , FILEHANDLESERVER , S4SYSTEMCARDALARM .  
Semantics : The NAP component category involved in reporting this event.

COMPONENTINFO Returns STRING  
Semantics : The full component information string for this event.

COMPONENTITEM Returns INTEGER mnemonic  
Mnemonic values : NDM , VMMM , COMS , NETUA , INTERP , TSPHIP , TSPTMK , TSPCAS , TSPPRIM , TSPSCCP , HEARTBEAT , TSPROUTER , CALLRECORD , PEPLIBRARY , AMISANALOG , CALLPROCESS , APPLUSERAGENT , TSPINTERPRETER , TSPSIGNALMANAGER , AIM , VSM , NIUM , SMDI , ADMIN , ROUTER , TSPDSP , TSPMTP , TSPTUP , TSPMAP , TSPISUP , TSPTCAP , TSPCACHE , TSPMODULE , TSPSS7MAN , NAPSUPPORT , VOICEFRAME , TSPGENERAL , LOGANALYZER , NAOPENAGENT , NAOPENSERVER , TSPFILESERVER , P24INTERPRETER .  
Semantics : The module identity of this component, if relevant.

COMPONENTNIU Returns INTEGER range 0 to 255  
Semantics : The NIU number involved in this event, if relevant.

COMPONENTNIULINK Returns INTEGER range 0 to 255  
COMPONENTVOICENIU is a synonym for this attribute  
Semantics : The NIU LINK number involved in this event, if relevant.

COMPONENTSUBCATEGORY Returns INTEGER range 0 to 16777215  
Semantics : Subcategory information about the NAP component involved in reporting this event.

COMPONENTSUBITEM Returns INTEGER range 0 to 16777215  
Semantics : The module subid of this component.

COMPONENTVCUNIT Returns INTEGER range 0 to 65535  
Semantics : The number of the Voice Control unit, if relevant.

COMPONENTVOICENIU Returns INTEGER range 0 to 255  
COMPONENTNIULINK is a synonym for this attribute  
Semantics : The VOICE NIU number involved in this event, if relevant.

EVENTCATEGORY Returns INTEGER mnemonic  
Mnemonic values : TSPEVENT , HOTSTANDBY , DIAGNOSTIC , APPLICATION , SYSTEMMSGDESK , CALLPROCESSING , VOICEPROCESSING , CRITICALRESOURCE , MAINTADMIN , DATACOMEVENT,, PROGRAMFAILURE , NETWORKINGEVENT .  
Semantics : The category assigned to this NAP log event.

EVENTRD Returns HEXADECIMAL STRING  
Semantics : The result descriptor associated with this event, if applicable.

EVENTREASON Returns INTEGER range 0 to 65535  
URC is a synonym for this attribute  
Semantics : The reason number assigned to this NAP log event.

EVENTSUBCATEGORY Returns INTEGER range 0 to 255  
Semantics : The subcategory assigned to this NAP event category.

JOBNUMBER Returns INTEGER range 0 to 65535  
Semantics : The JOB number of the program logging this NAP event.

LINE Returns STRING  
Parameters : 1. Integer range 0 to 255  
Semantics : Text of an individual line in LOGTEXT as referenced by parameter.

LINENUMBER Returns INTEGER range 0 to 268435455  
Semantics : The line number at which the event was reported by this NAP module.

LOGTEXT Returns STRING  
Semantics : The analyzed text of the NAP log entry as decoded by the NAPLOGANALYZER utility.

MIXNUMBER Returns INTEGER range 0 to 65535  
Semantics : The MIX number of the program logging this NAP event.

MODULEID Returns INTEGER mnemonic  
Mnemonic values : NDM , VMMM , COMS , NETUA , INTERP , TSPHIP , TSPTMK , TSPCAS , TSPPRIM , TSPSCCP , HEARTBEAT , TSPROUTER , CALLRECORD , PEPLIBRARY , AMISANALOG , CALLPROCESS , APPLUSERAGENT , TSPINTERPRETER , TSPSIGNALMANAGER , AIM , VSM , NIUM , SMDI , ADMIN , ROUTER , TSPDSP , TSPMTP , TSPTUP , TSPMAP , TSPISUP ,

TSPTCAP , TSPCACHE , TSPMODULE , TSPSS7MAN , NAPSUPPORT ,  
 VOICEFRAME , TSPGENERAL , LOGANALYZER ,  
 NAOPENAGENT , NAOPENSERVER , TSPFILESERVER , P24INTERPRETER .  
 Semantics : The identity of the reporting NAP module

**MODULESUBID** Returns INTEGER range 0 to 255  
 Semantics : The sub-identity of the reporting NAP module.

**PARAMS** Returns INTEGER range 0 to 255  
 Semantics : The number of additional parameters (ADDLPARAMS).

**REASONTEXT** Returns STRING  
 Semantics : Returns a brief text description from the NAPLOG describing the URC

**REVISIONLEVEL** Returns INTEGER range 0 to 255  
 Semantics : The revision level of the codefile associated with the calling module.

**SEVERITY** Returns INTEGER range 0 to 255  
 Semantics : The SEVERITY code of this NAP log entry.

**SEVERITYTEXT** Returns INTEGER mnemonic  
 Mnemonic values : MAJORFAILURE , LOSTVOICEDATA , DELIBERATEABORT ,  
 SINGLECALLTERMINATE , NAPNORMAL ,  
 MINORFAILURE , NAPINOPERABLE , NOCALLSAFFECTED ,  
 UNINTENDEDABORT , MULTICALLTERMINATE .  
 Semantics : The SEVERITY CODE of the NAP log entry.

**TIMESTAMP** Returns STRING as "HH:MM:SS"  
 Semantics : Returns the date and timestamp of the NAP log event.

**TOTALLINES** Returns INTEGER  
 Semantics : The number of lines generated in LOGTEXT attribute.

**URC** Returns INTEGER range 0 to 65535  
 EVENTREASON is a synonym for this attribute  
 Semantics : The reason number assigned to this NAP log event.