NonStop NET/MASTER Tips and Techniques

by John New
Gresham Software Labs
Email: jnew@greshamsoftwarelabs.com.au

This article originally appeared in *The Tandem Connection*, Volume 18, No. 4 - September 1997, and is reproduced with permission from the International Tandem Users' Group (ITUG).

Introduction

This is an ongoing column with NonStop NET/MASTER tips and techniques. Each column is also accessible from http://www.greshamsoftwarelabs.com.au/ (Gresham Software Labs). Please send all comments and suggestions to John New at jnew@greshamsoftwarelabs.com.au.

Biography

John New is a technical writer. He has written and updated various Tandem manuals. He currently writes hard-copy, online, and web documents for a variety of software products.

==========

NonStop NET/MASTER MS Tips and Techniques
Solving the Problem of Missing Device Names

==========

Introduction

This column discusses some of the difficulties caused by the problem of missing device names in EMS event messages generated by certain software products. It describes how NonStop NET/MASTER Rule Management Services (RMS) solves this problem for local console event messages. It explains how you can customize RMS to solve the problem for all event messages.

The Problem of Missing Device Names

Devices on a Tandem system, which are called logical devices (LDEVs), have a number and name. The LDEV number is assigned at system generation (for example, $6) and uniquely identifies a device. The LDEV name is the corresponding name of the device (for example, $SYSTEM). (An SCF LISTDEV command provides a list of local devices, which includes their LDEV numbers and names.)

Many event messages report information about devices. Messages are generated, for example, when a device is started or aborted. The information may include the LDEV number, the LDEV name, or both. An event message may contain LDEV number and name information in the displayable message text or elsewhere in the message.

Note: Information in an event message is contained in Subsystem Programmatic Interface (SPI) tokens. If present, the message text is contained in the ZEMS-TKN-TEXT token, the LDEV number in ZEMS-TKN-LDEV, and the LDEV name in ZEMS-TKN-LDEVNAME.

However there are sometimes inconsistencies in how event messages from certain software products deliver information. Event messages that have an LDEV number but no LDEV name can present difficulties for all Tandem users, particularly operators (who are perhaps responsible for many systems in a network, possibly with one system working as a central node collecting messages from other remote nodes).

First, LDEV numbers can change with each SYSGEN. This makes it difficult to automate operations for those messages that do not contain LDEV names. Second, it may be time-consuming for an operator in a large network with a lot of message traffic to determine the LDEV name that corresponds to an LDEV number. This may delay the resolution of a problem. Finally, events that need operator action are not very helpful if only the number is displayed in the message text. The operator must determine the LDEV number and the device type before being able to decide what action is needed.

A software product that monitors all events and acts intelligently to make event messages consistent would be useful for Tandem users. Such software could help operators by enhancing event messages to display an LDEV name instead of, or in addition to, an LDEV number.

The Solution for Local Console Event Messages

RMS is distributed with a solution to the problem of missing device names for local console event messages (local messages from the EMS subsystem in the range EMS0001 through EMS0511, which start with the word LDEV). RMS automatically does some initial processing on these messages before the messages are passed on for further processing. RMS translates the LDEV number into the LDEV name and uses the LDEV name as the subject of the message.

Note: The displayable message text is not changed. However, if a console event message triggers a rule, you could use RMS to modify the message text to include the LDEV name. The &zzzmsubject RMS message variable contains the subject of a message.

(RMS is a NonStop NET/MASTER Management Services (MS) application. RMS uses rules to perform its functions, such as automating operations. A rule, which is triggered by criteria such as a certain message, is a set of instructions to RMS to perform certain tasks. A ruleset is a set, or collection, of rules. An active message handler runs with a ruleset, and processes messages according to the rules in the ruleset. You can find out more about RMS by reading the NonStop NET/MASTER RMS Management and Operations Guide.)

A Complete Solution for All Event Messages

You can use RMS to solve the problem of missing device names for all event messages. That is, you can customize RMS to also handle messages from remote systems, local messages that are not from the EMS subsystem, and local messages from the EMS subsystem equal to or greater than EMS0512.

A complete solution to the problem of missing LDEV names could include a package with two components: an RMS preprocessing exit; and an external Guardian utility that converts LDEV numbers to names. The programs in the package would detect and process all event messages with "LDEV" followed by a number in the message text.

Note: If you are already using a preprocessing exit and you also want to solve the problem of missing LDEV names, you may need to merge the functionality of your existing preprocessing exit with the functionality described in this column.

Without such a preprocessing exit, the following messages, with only an LDEV number, might be displayed after a CMI START LINE command:

```
EMS0141 LDEV 0027 CU %030 CLIP DOWNLOADED
EMS0006 LDEV 0027 CU %030 UP
EMS0006 LDEV 0027         UP
```

After installing the preprocessing exit, the message text could change to display just the LDEV name, for example:

```
EMS0141 DEVICE \TEST.$ATP1 CU %030 CLIP DOWNLOADED
EMS0006 DEVICE \TEST.$ATP1 CU %030 UP
EMS0006 DEVICE \TEST.$ATP1 UP
```

Or, more usefully, perhaps both the LDEV name and number:

```
EMS0141 DEVICE \TEST.$ATP1 LDEV 0027 CU %030 CLIP DOWNLOADED
EMS0006 DEVICE \TEST.$ATP1 LDEV 0027 CU %030 UP
EMS0006 DEVICE \TEST.$ATP1 LDEV 0027         UP
```

The Preprocessing Exit NCL Procedure - LDEVEXIT

An RMS preprocessing exit is an NCL procedure that processes messages before RMS rule processing begins. Using NCL procedures before RMS rule processing allows more efficient message filtering and global alteration of messages. Such an NCL procedure supplements other RMS message processing. (You specify the NCL procedure name in the Pre-Processing NCL Exit field on the RMS : Control Options Definition-Suppression panel.)

For convenience, we will assume that we are designing a preprocessing exit called LDEVEXIT and that LDEVEXIT calls a function called NUM2NAME.

LDEVEXIT could perform some initial message filtering (messages without SPI tokens and without "LDEV" in the text would not be processed) and could check that the value following "LDEV" is a valid number. For example:

```
/* Check for SPI tokens */
if &$&(msg).spi = '' then exit

/* Check for "LDEV" in message text */
&msgtext = &$&(msg).text
&pos = wordpos( 'LDEV', &msgtext)
if &pos = 0 then exit

/* Make sure ldev number is a number */
if &ldev_num = '' then exit
if not datatype( &ldev_num, 'N' ) then exit
```

If LDEV is a valid number, the originating node name would be added to the LDEV number and the entire value would be passed to an NCL function called NUM2NAME for LDEV number-to-name resolution. For example:

```
/* Add node name to ldev number */
&ldev_num = &$&(msg).spi.tandem.zems_tkn_nodename||'.$'||&ldev_num

/* Get ldev name */
&ldev_name = num2name( &ldev_num )
if &ldev_name = '' then
    /* Ldev name could not be retrieved */
    exit
```

After NUM2NAME completes, LDEVEXIT could insert the LDEV name into the token ZEMS-TKN-LDEVNAME, this token could become the subject of the event, and the message text could be changed to display the LDEV name rather than the LDEV number or both the name and number. For example:

```
/* Insert ldev name into event token */
&$&(msg).spi.tandem.zems_tkn_ldevname = &ldev_name

/* Point subject mark token to ldevname token to make
   the ldevname the subject of the event              */
&$&(msg).spi.tandem.zems_tkn_subject_mark =,
 'TANDEM.ZEMS_TKN_LDEVNAME{1}'

/* Change event text to display just ldev name */
&newtext = left( &msgtext, wordindex( &msgtext, &pos ) - 2 )
&newtext = &newtext 'DEVICE' &ldev_name
&newtext = &newtext subword( &msgtext, &pos + 2 )
&$&(msg).text = &newtext

/* Change event text to display both ldev name and number */
&charpos = wordindex( &msgtext, &pos )
&newtext = left( &msgtext, &charpos - 2 )
&newtext = &newtext 'DEVICE' &ldev_name
&newtext = &newtext substr( &msgtext, &charpos )
&$&(msg).text = &newtext
```

NUM2NAME would determine whether the LDEV number has been previously resolved to an LDEV name (to reduce network traffic, it could maintain a list of previously resolved LDEV names in memory). If so, it would pass the LDEV name back to LDEVEXIT. If not, it would pass the LDEV number to an external Guardian utility for LDEV number-to-name resolution (for convenience, we will call the utility ISSCOM). For example:

```
/* First check whether we have this one in memory */
&ldev_name = &$uvldev.&ldev_num
if &ldev_name \= '' then
    do
    /* Return saved ldev name */
    return( &ldev_name )
    end

/* Send command to ISSCOM utility */
intcmd 'isscom ldev' &ldev_num
```

After obtaining the LDEV name from the utility, NUM2NAME could store the resulting LDEV name in memory and would pass the LDEV name back to LDEVEXIT.

```
/* Store converted ldev for later use */
&$uvldev.&ldev_num = &ldev_name

/* Pass back ldev name */
return( &ldev_name )
```

The External Guardian Utility - ISSCOM

The external Guardian utility, ISSCOM, would do the job of LDEV number-to-name conversion. It would be capable of resolving local and remote LDEV numbers (from remote systems connected by Expand). After resolution, ISSCOM would pass the LDEV name to NUM2NAME.

Ideally, ISSCOM would be a command interpreter that you could also invoke interactively from TACL or the OCS command input line. Being able to easily determine the LDEV name that corresponds to an LDEV number would be very useful for all Tandem users. There are many ways to obtain an LDEV name on your local system. To obtain the LDEV name that corresponds to an LDEV number on a remote system, you would be able to type a command such as the following using ISSCOM from TACL:

```
ISSCOM LDEV \SYS1.$6
```

Or the following OPSYS command from OCS:

```
OPSYS SEND ISSCOM LDEV \SYS1.$6
```

In either case, the result would be the same, for example:

```
LDEV name: \SYS1.$SYSTEM
```

Before being able to invoke ISSCOM from OCS, you would have to define ISSCOM as an external Guardian utility to NonStop NET/MASTER MS. You would use the following definition from the UMS : Utility Details panel:

```
Utility Name ........  ISSCOM
OPSYS or PROGRUN ....  O
File/Macro Name .....  $SYSTEM.SYSTEM.ISSCOM
Startup Text ........
TACL Macro? .........  N
Reads IN File? ......  Y
```

To make ISSCOM available to all NonStop NET/MASTER MS users you would specify a low authority level from the UMS : Utility Command Set Details panel:

```
Command         Authority
-------------------------
*                     0
```

Other Options

If you are using a preprocessing exit and you do not want to modify all event messages with SPI tokens and with "LDEV" in the text (just messages from certain subsystems, or with certain numbers, or within a certain range of numbers), you could tailor LDEVEXIT to process only specified messages during preprocessing. Initial filtering in the preprocessing exit by subsystem and message number could offer performance improvements.

A message validation exit NCL procedure provides a final check on whether a message triggers an RMS rule. If you want to modify only event messages with SPI tokens and with "LDEV" in the text that have already satisfied all other rule trigger criteria, you could use LDEVEXIT as a message validation exit. (You specify the NCL procedure name in the User Exit field on the second RMS : Message Validation panel.)

Conclusion

Missing LDEV device names in event messages can present problems for all Tandem users, particularly operators. RMS solves the problem for local console event messages. You can easily customize RMS to solve the problem for all event messages.

Note: If you want to try out the techniques described in this column, you can download some free sample NCL code (LDEVEXIT and NUM2NAME) and a free sample command interpreter (ISSCOM) from the Gresham Software Labs web site (http://www.greshamsoftwarelabs.com.au/).