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 17, No. 5 - October/November 1996, 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 Tips and Techniques
Using RMS to Automatically Retry Rule Actions

This column discusses a NonStop NET/MASTER Rule Management Services (RMS) feature that allows you (the Network Control Language (NCL) programmer) to write or modify rule action NCL procedures that automatically retry rule actions.

Note: You can find out more about RMS by reading the NonStop NET/MASTER RMS Management and Operations Guide.

Background Information

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. RMS is distributed with two standard rulesets (BASERULE and STATSCAP). An active message handler runs with a ruleset, and processes messages according to the rules in the ruleset.

The actions performed by a rule are often implemented using NCL procedures. For example, the ZCPUDOWN rule, which is part of the BASERULE ruleset distributed with RMS, is triggered by event message 101 from the CPU subsystem. The actions performed by this rule are implemented by the rule action NCL procedure called ZRMSCPUN.

You can code a rule action NCL procedure to attempt to perform actions once and report that the actions either succeeded or failed; or, if the actions fail, you can code the NCL procedure to attempt to automatically retry the actions. Using a rule action NCL procedure to automatically retry actions can simplify the responsibilities of system operators.

For example, ZRMSCPUN automatically dumps the memory of a failed CPU and reloads the CPU. If the reload fails, ZRMSCPUN is coded to retry the reload every 60 seconds rather than requiring operator intervention to attempt periodic reloads.

Coding NCL procedures to automatically retry rule actions also allows RMS to remove the NCL procedure from memory during the retry delay period, which frees system resources.

An NCL procedure typically indicates the success or failure of actions by assigning a return code value to the &SYS.RETCODE system variable. A rule action NCL procedure indicates whether an automatic retry is required by assigning one of the following values to &SYS.RETCODE:

0	Indicates that the rule action completed successfully and a retry should not be performed.
4	Indicates that the rule action failed and a retry should be performed after a delay (the default 60 seconds).
16	Indicates that the rule action failed but a retry should not be performed.

For example, after attempting to reload a CPU, the code that attempts to reload the CPU in ZRMSCPUN assigns one of the following values to &SYS.RETCODE:

0	Indicates that the CPU reload was successful.
4	Indicates that CPU recovery failed and the rule is to be retried.
16	Indicates that CPU recovery failed but returned to RMS if CPU recovery fails and retries expired.

You can use a variety of core statements and verbs to set &SYS.RETCODE. Core statements that can set the variable include CALL, EXIT, and RETSUB; verbs include LOCK, PANEL, SECCALL, and START. Typically, you would use either EXIT to explicitly set a value in &SYS.RETCODE when a recovery NCL procedure terminates, or RETSUB to set a value when a recovery subroutine (called by a GOSUB core statement) finishes.

Note: If you do not explicitly set the return code, &SYS.RETCODE may contain a value that does not correctly indicate the result of the rule action.

Other variables you can use when writing or modifying rule action NCL procedures to implement automatic rule retry are:

&SYSMSG	This variable contains the message text, which you specify, to indicate the success or failure of a recovery operation.
&$RETRY	This read-only variable indicates the number of rule action retries attempted, including the current attempt.
&$RETRYDELAY	This variable contains the period of delay (in seconds), which you specify, before a rule action is retried.

You can display or delete rule action NCL procedures that are scheduled for retry after a delay by selecting the appropriate option from the RMS : Message Action Control menu panel. You can also use the RMS DISPLAY DELAYED and RMS DELETE DELAYED commands, respectively, from the Operator Control Services command input line.

Examples

The following code segment shows how to attempt a recovery action and then perform a simple test of the result. A subroutine (recover_subject) attempts to recover a subject. A test (if ... then ... else) determines the action to take after the attempt.

If the attempt succeeds, the NCL procedure exits with &SYS.RETCODE set to 0. If the attempt fails, the NCL procedure exits with &SYS.RETCODE set to 4 and specifies that a retry should be performed after 90 seconds.

```
...
/* Subroutine returns 0 or 16 */
gosub recover_subject
/* Did recovery succeed or fail? */
if &sys.retcode = 0 then
    /* Recovery succeeded */
    exit 0
else
    /* Recovery failed */
    /* Retry after 90 seconds */
    do
    &$retrydelay = 90
    exit 4
    end
...
```

The following code segment shows how to attempt a recovery action and then perform a complex test of the result. A subroutine (recover_subject) attempts to recover a subject. A test (if ... then ... else ... select) determines the action to take after the attempt.

If the attempt succeeds, the NCL procedure exits with &SYS.RETCODE set to 0. If the attempt fails, the retry delay period depends on the number of retries; the greater the number of retries, the longer the retry delay. This technique gradually slows down recovery attempts. When the number of retries exceeds 40, recovery attempts are abandoned.

The &ZZZMSUBJECT RMS variable contains the name of the subject that the NCL procedure is attempting to recover.

```
...
/* Subroutine returns 0 or 16 */
gosub recover_subject
/* Did recovery succeed or fail? */
if &sys.retcode = 0 then
    /* Recovery succeeded */
    do
    &sysmsg = "Device "&zzzmsubject" recovered"
    exit 0
    end
else
    /* Recovery failed */
    select
        when &$retry <= 5 then
            /* Retry after 30 seconds */
            do
            &$retrydelay = 30
            exit 4
            end
        when &$retry > 5 and &$retry <= 10 then
            /* Retry after 60 seconds */
            do
            &$retrydelay = 60
            exit 4
            end
        when &$retry > 10 and &$retry <= 20 then
```

```
            /* Retry after 300 seconds */
            do
            &$retrydelay = 300
            exit 4
            end
        when &$retry > 20 and &$retry <= 40 then
            /* Retry after 3600 seconds */
            do
            &$retrydelay = 3600
            exit 4
            end
        otherwise
            /* Retries expired */
            do
            &sysmsg "Recovery of "&zzzmsubject" failed, retries expired"
            exit 16
            end
    end
...
```

Note: NCL procedures you write or modify are normally placed in the customized NCL procedure library (ZNNMNCS).