CA Mainframe Network Management

NetMaster REXX Guide

r11.7

CA Product References

This document references the following CA products:

- CA NetMaster® Network Management for TCP/IP (CA NetMaster NM for TCP/IP)
- CA NetMaster® Network Management for SNA (CA NetMaster NM for SNA)
- CA NetMaster® Network Automation (CA NetMaster NA)
- CA SYSVIEW® Performance Management (CA SYSVIEW)
- CA 7 Workload Automation (CA 7 WA)
- CA Scheduler Job Management (CA Scheduler JM)

Contact CA

Contact Technical Support

For your convenience, CA provides one site where you can access the information you need for your Home Office, Small Business, and Enterprise CA products. At http://ca.com/support, you can access the following:

- Online and telephone contact information for technical assistance and customer services
- Information about user communities and forums
- Product and documentation downloads
- CA Support policies and guidelines
- Other helpful resources appropriate for your product

Provide Feedback

If you have comments or questions about CA product documentation, you can send a message to techpubs@ca.com.

If you would like to provide feedback about CA product documentation, complete our short customer survey, which is also available on the CA Support website, found at http://ca.com/docs.

Contents

Chapter 1: Introducing NetMaster REXX

This section contains the following topics:

About This Guide

This guide contains information for experienced REXX programmers who want to use REXX in a CA NetMaster environment. It provides information about any differences between NetMaster REXX and IBM's REXX to help you maintain and write REXX procedures. It does not tell you how to write the procedures. It also describes facilities that help you migrate your existing REXX procedures to work in the CA NetMaster environment.

NetMaster REXX

NetMaster REXX lets you write and execute REXX programs and procedures in the CA NetMaster environment. This includes REXX programs written specifically for CA NetMaster, as well as existing REXX programs written to run under IBM Tivoli NetView for z/OS (Tivoli NetView).

Writing and executing REXX programs and procedures in the CA NetMaster environment lets you do the following:

- Customize your product without learning the Network Control Language (NCL).
- Write specific monitoring and control procedures in a familiar language.
- Execute REXX programs written to execute in the Tivoli NetView REXX environment.

CA NetMaster is *not* Tivoli NetView, and NetMaster REXX is *not* Tivoli NetView REXX; however, CA NetMaster supports emulation of Tivoli NetView facilities. This can be summarized as follows:

- In this guide, the term *NetView REXX* means a REXX executable that is written to execute under Tivoli NetView
- By prefixing a NetView REXX command with NV (and a space), you can execute the command anywhere a NetMaster REXX command can be entered (including OCS, Command Entry, Operator Console Modify commands, NCL procedures, and so on).
- By issuing the NCCF command at the OCS window command prompt, you can enter an NCCF Emulation mode, whereby all following commands entered at the OCS screen are treated as NetView REXX commands.

Migration from Tivoli NetView

The provided REXX support can ease migration from Tivoli NetView; however, the REXX support is not identical to Tivoli NetView, so migration cannot be accomplished without effort. In some cases, you must modify or rewrite existing NetView REXX.

Because Tivoli NetView is emulated, differences in program response times may be apparent. If you use complex PIPE commands, you should run them in CA NetMaster to evaluate whether there are any specific performance issues. To address performance issues, you may need to rewrite procedures or replace them with the code-free facilities of CA NetMaster.

NetMaster REXX and TSO/E REXX

In z/OS, the typical and most familiar implementation of REXX is the TSO/E REXX interpreter. This interpreter is also used in other IBM products and environments such as Tivoli NetView.

The REXX implementation used in CA NetMaster is called NetMaster REXX. It uses a REXX engine called GREXX. GREXX is a REXX implementation written by CA.

GREXX is a compiled implementation of REXX. Because GREXX compiles rather than interprets source, there are differences between the GREXX and TSO/E implementations of REXX. The differences and similarities are covered in detail in this chapter.

Libraries and Source Format

TSO/E REXX and NetMaster REXX store the REXX source in libraries known as partitioned data sets. While TSO/E REXX allows the libraries to have various record lengths and formats, NetMaster REXX requires the libraries to have a fixed length of 80 byte records (blocking is allowed).

The source can be numbered. Both implementations use the convention of examining the first line of the source. If columns 73-80 are numeric, the member is regarded as numbered, and only columns 1–72 are inspected for valid source. If columns 73–80 are not all numeric, the member is not regarded as numbered, and all columns, including 73-80, are regarded as source.

TSO/E REXX normally requires the source libraries to be allocated to the SYSPROC or SYSEXEC ddname concatenation. NetMaster REXX normally requires the source libraries to be in the COMMANDS ddname concatenation. However, this can be overridden for a specific user through the NCL Library DDNAME UAMS attribute.

Tivoli NetView REXX requires the libraries to be allocated to a specific ddname, DSICLD.

REXX Source Recognition

TSO/E and Tivoli NetView recognize REXX as distinct from CLIST, by requiring a comment on the first line of a procedure that includes the characters REXX.

CA NetMaster also requires a comment in the first line, to enable it to distinguish REXX source from NCL (and REXX object), but with the following additional rules:

- The first source line must begin with a REXX comment (that is /*) as the first non-blank characters on the line.
- The first non-blank characters after the opening comment must be REXX (case is unimportant). There must be no embedded blanks in these characters and these characters must be on the first line.

Compile and Save in Object Form

The NetMaster REXX implementation lets you compile and save REXX programs in object form. The generated object is also stored in libraries in FB/80 format. This allows the object to coexist in the same concatenation (COMMANDS) as source. Standard search order is used to locate a member. The NetMaster REXX implementation recognizes the object at load time.

Language and Implementation Differences

The following sections describe the differences between TSO/E REXX and NetMaster REXX. They follow the same order as IBM's *TSO/E REXX Reference*. The following chapters from that book are referenced:

- Chapter 2 - REXX General Concepts
- Chapter 3 - Keyword Instructions
- Chapter 4 - Functions
- Chapter 5 - Parsing
- Chapter 6 - Numbers and Arithmetic
- Chapter 7 - Conditions and Condition Traps

Generally, with some exceptions, only differences are mentioned.

Note: The NetMaster REXX implementation was designed to be compliant with the REXX language as described in the book, *The REXX Language: A Practical Approach to Programming - Second Editio*n, by Mike Cowlishaw (Prentice Hall).

NetMaster REXX Structure and General Syntax

Relevant issues include:

- The standard EBCDIC character set is always used for NetMaster REXX source.
- Double-Byte Character Support (DBCS) is not supported.
- Nested comments are supported by NetMaster REXX.
- Normal, hexadecimal, and binary strings are supported by NetMaster REXX. Quoted strings that extend across source lines are not normally supported.
- Tokens, symbols, and so on are supported identically.

The normal REXX language definition does not accept quoted strings that cross source lines. The IBM TSO/E REXX interpreter has supported this for many years; however, the support is not consistent, and trailing blanks are not always accepted. The length of the entire quoted string is still limited to 250 characters.

NetMaster REXX does not accept quoted strings that cross source lines. However, it provides an option to enable REXX source containing this construct to mostly compile.

More information:

Libraries and Source Format (see page 13)
REXX Source Recognition (see page 14)
RXCTL (see page 29)
RXCHECK (see page 28)

Expressions and Operators

The difference between NetMaster REXX and TSO/E REXX expressions is the limit on their length, intermediate value, or result, as follows:

- TSO/E REXX limits values (intermediate or final) to 16,777,215 characters.
- NetMaster REXX limits values (intermediate or final) to 32,000 characters.

Note: This is the character representation of the value. (Remember that REXX is a typeless language. All values are stored in character form.) It does not limit the mathematical range or value, as long as the value can be represented in 32,000 characters.

The operators /= and /== are supported in TSO/E REXX as alternatives to \= and \==, respectively. These are not part of the proposed REXX standard and are not supported by GREXX. Use backslash (\) or not (¬) instead of slash (/).

Clauses and Instructions

There are no specific differences here between NetMaster REXX and TSO/E REXX.

Assignments and Symbols

There are no specific differences here between NetMaster REXX and TSO/E REXX.

Note: The assigned value of a variable cannot exceed 32,000 characters in length.

Commands to External Environments

The NetMaster REXX implementation supports some of the external environments supported by TSO/E REXX and NetView REXX. It also supports some environments specific to CA NetMaster.

More information:

ADDRESS Environments (see page 33)

Keyword Instructions

This section describes each REXX keyword instruction and lists the NetMaster REXX differences.

ADDRESS

There are no specific NetMaster REXX issues with the ADDRESS instruction.

More information:

ADDRESS Environments (see page 33)

ARG

There are no specific NetMaster REXX issues with the ARG instruction.

More information:

PARSE (see page 20)
Parsing (see page 25)

CALL

The CALL instruction is supported. This includes CALL ON/OFF condition support. In addition, NetMaster REXX has an extension for a standard CALL.

If the name of the program being called is specified in parentheses, then the name is treated as a simple variable, and the actual program name to be called is the value of that variable. This allows you to eliminate many uses of the INTERPRET statement.

For example, in REXX, to call a program that is dynamically named:

```
PROGRAM = <some name>
INTERPRET 'CALL 'PROGRAM' arg,arg,…'
```

In NetMaster REXX, you can code instead:

```
PROGRAM = <some name>
CALL (PROGRAM) arg, arg, …
```

DO

There are no specific NetMaster REXX issues with the DO instruction.

DROP

There are no specific NetMaster REXX issues with the DROP instruction.

EXIT

There are no specific NetMaster REXX issues with the EXIT instruction. The Return Value that is displayed at the end of a NetMaster REXX process must be a valid number that is in the range: -2**32 to 2**32-1. If not, it is ignored.

IF

There are no specific NetMaster REXX issues with the IF instruction.

INTERPRET

NetMaster REXX supports the INTERPRET instruction; however, because NetMaster REXX is a compiler, supporting INTERPRET involves significant overhead. The entire NetMaster REXX compiler must be used to compile the value of the expression (after evaluation); therefore, use of INTERPRET is very expensive.

We recommend that you avoid using INTERPRET wherever possible. For example, refer to the CALL (see page 17) statement for one NetMaster REXX extension that can eliminate the use of INTERPRET.

ITERATE

There are no specific NetMaster REXX issues with the ITERATE instruction.

LEAVE

There are no specific NetMaster REXX issues with the LEAVE instruction.

NOP

There are no specific NetMaster REXX issues with the NOP instruction.

NUMERIC

The NUMERIC instruction is supported with the following restrictions:

- NUMERIC FORM is not supported. A compiler error occurs if a numeric form is found in the source. NetMaster REXX always behaves as if NUMERIC FORM SCIENTIFIC is in effect.
- NUMERIC DIGITS and NUMERIC FUZZ require an operand. To reset to the default value, you must specify that value. (For example, NUMERIC DIGITS 9 or NUMERIC FUZZ 0.)

OPTIONS

The OPTIONS instruction is used to provide environment-specific extensions to REXX. It is followed by an expression that is analyzed by the implementation.

In NetMaster REXX, the OPTIONS expression is ignored at compile time. At execution time, the expression is evaluated and broken into blank-delimited tokens. These tokens are then processed as described in the following table.

Note: The REXX language definition says that unrecognized OPTIONS operands should be ignored. Generally, this is done. However, specification of some operands in REXX code that normally executes under the TSO/E REXX interpreter is probably an indication of problems ahead. For example, use of keywords that imply support for Double Byte Character Sets.

The following table lists recognized tokens from the OPTIONS instruction expression. All other tokens are ignored.

Token	Response
NOETMODE	Ignored
NOEXMODE	Ignored
ETMODE	Error raised (no DBCS support)
EXMODE	Error raised (no DBCS support)
NOEMULSAY	Disables REXXHELP processing of SAY statements (internal use only)
EMULSAY	Enables REXXHELP processing of SAY statements (internal use only)
NOEMULTRACE	Disables REXXHELP processing of TRACE statements (internal use only)
EMULTRACE	Enables REXXHELP processing of TRACE statements (internal use only)
NOEXECDUMP	No special NetMaster REXX termination dump (internal use only)
EXECDUMP	Special NetMaster REXX termination dump at process end (internal use only)
MAXSTOR=*n*K	Overrides the SYSPARM-set max storage default

PARSE

The PARSE instruction is fully supported by NetMaster REXX with the following differences:

- PARSE EXTERNAL is not presently supported. (This is a TSO/E REXX extension.) You can emulate PARSE EXTERNAL with PARSE PULL when the stack is empty.
- PARSE PULL reads from the stack (per TSO/E REXX). If the stack is empty, a message is sent to the environment (the terminal) requesting input. The GO command can be used to supply input to the waiting procedure. (This is similar to the implementation of PARSE PULL in Tivoli NetView.)
- PARSE NUMERIC is not supported. The three values used as input to the parsing template are the current settings of NUMERIC DIGITS, NUMERIC FUZZ, and NUMERIC FORM. The first two (DIGITS and FUZZ) can be obtained through built-in functions. The last will always be SCIENTIFIC in NetMaster REXX.
- PARSE SOURCE provides, as input to the parsing template, a string in the following format:

  ```
  GREXX COMMAND PMTGREXX COMMANDS ? PMTGREXX NM NM NM
  ```

  TSO/E REXX provides a string of the format:

  ```
  TSO COMMAND PMTGREXX SYS00271 AUDE0.LIB.NCL ? TSO ISPF ?
  ```

- PARSE VERSION provides, as input to the parsing template, a string in the following format:

  ```
  REXX/CA 3.92 18 Oct 1995
  ```

  TSO/E REXX provides a string in the format:

  ```
  REXX370 3.48 01 May 1992
  ```

The string provided by PARSE SOURCE consists of the following tokens:

- The string GREXX (IBM provides TSO).
- The string COMMAND, FUNCTION, SUBROUTINE, or SERVER indicating how the procedure was invoked. (Currently, NetMaster REXX always uses SUBROUTINE for a called procedure. It does not distinguish function calls.)
- The name of the procedure in uppercase.
- The library ddname (normally COMMANDS).
- A question mark (in TSO/E REXX, possibly the data set name that the procedure was loaded from).
- The procedure name again.
- Initial (default) command environment (normally NM).
- Address Space name (normally NM).
- Parsetok (normally NM).

PROCEDURE

There are no specific NetMaster REXX issues with the PROCEDURE instruction.

PULL

There are no specific NetMaster REXX issues with the PULL instruction. Variable values (and stack records) in NetMaster REXX are limited to 32000 characters.

PUSH

There are no specific NetMaster REXX issues with the PUSH instruction. Variable values (and stack records) in NetMaster REXX are limited to 32000 characters.

QUEUE

There are no specific NetMaster REXX issues with the QUEUE instruction.Variable values (and stack records) in NetMaster REXX are limited to 32000 characters.

RETURN

There are no specific NetMaster REXX issues with the RETURN instruction.

For comments regarding the return value from a top-level procedure, see EXIT (see page 17) instruction.

SAY

There are no specific NetMaster REXX issues with the SAY instruction.

In NetMaster REXX, the destination for the message produced by the SAY instruction is the environment that the NetMaster REXX procedure is executing in. For example, if executed from OCS, the destination is the OCS window. If executed in a background region, for example BSYS, the log is the destination.

SELECT

There are no specific NetMaster REXX issues with the SELECT instruction.

SIGNAL

There are no specific NetMaster REXX issues with the SIGNAL instruction.

Note: All labels referenced in a SIGNAL statement including those implied (for example, SIGNAL ON NOVALUE) must exist. This applies even if the statement is not executed. The compiler checks this, and missing labels cause a compilation failure.

More information:

Conditions and Condition Traps (see page 25)

TRACE

The NetMaster REXX implementation of the TRACE instruction has the following differences:

- TRACE S (SCAN) is not supported.
- The prefix options (?, And, !) are not supported.
- Interactive debug (TRACE ?) is not supported.

UPPER

There are no specific NetMaster REXX issues with the UPPER instruction.

Built-in Function Support

The following describes the similarities and differences in built-in function support in NetMaster REXX.

Generally, functions are supported identically in NetMaster REXX. This includes the use of internal definitions to replace built-ins, quoted strings as function names, and so on.

The actual list of built-in functions supported by NetMaster REXX is almost identical to TSO/E REXX. Because there are many built-in functions, the following sections list only those that are unsupported, or that have significant issues.

Note: External functions, such as those provided by TSO or Tivoli NetView, are not explained in the following section.

Built-in Function	Description
CONDITION	S (Status) is not supported. I (Instruction) always returns SIGNAL.
DB*xxxx*	Because NetMaster REXX does not support DBCS, none of the DB functions are supported.
ERRORTEXT	Ranges 64 to 99 will return NetMaster REXX extended error message.
EXTERNALS	This is not provided by NetMaster REXX. It causes a compilation error if used in source.
FORMAT	NetMaster REXX does not support the expp and expt arguments. Only a maximum of three arguments are allowed.
LINESIZE	NetMaster REXX always returns 80.
SOURCELINE	Only source lines containing REXX code are retained after compilation. Lines that contain only comments are treated as blank.
STORAGE	NetMaster REXX does not allow use of the STORAGE function.

Built-in Function	Description
TIME	TIME(R) does not reset the elapsed-time clock.
TRACE	The F and S options are not supported, nor are the ! and ? prefixes. **Note:** F (Failure) means the same as N (Normal).

More information:

REXX External Assembler API (see page 83)

TSO/E External Functions

The following functions listed in the *TSO/E REXX Reference* as TSO/E external functions are not supported by NetMaster REXX:

- GETMSG
- LISTDSI
- MSG
- MSGVAR
- OUTTRAP
- PROMPT
- SETLANG
- STORAGE
- SYSCPUS
- SYSDSN
- SYSVAR

Parsing

This section describes differences between TSO/E REXX and NetMaster REXX parsing, as implemented in the ARG (see page 16), PARSE (see page 20), and PULL (see page 21) instructions.

NetMaster REXX handles all parsing template options as documented by TSO/E REXX.

The following are some considerations:

- Relative positional patterns (for example, +35) must have no spaces between the sign and the number.
- If breaking a parsing template across source lines, leave a blank after the last template specification, before the comma that signifies continuation to the next line.

Numbers and Arithmetic

NetMaster REXX supports numbers with arbitrary precision, as does TSO/E REXX.

For efficiency, it is best if NUMERIC DIGITS 9 is in effect. Larger values require the use of slower arithmetic routines. Smaller values do not speed things up.

NetMaster REXX follows the standard REXX rules about preserving trailing zeros and so on.

NetMaster REXX does not support the NUMERIC FORM statement.

Conditions and Condition Traps

NetMaster REXX supports conditions and condition traps, with the following considerations:

- Labels for referenced condition names must be defined. For example, SIGNAL ON NOVALUE implies a reference to a label named NOVALUE. If this label is not defined in the source procedure, a compilation error occurs. (If the NAME option is used on a SIGNAL ON or CALL ON instruction, then the specified label name must exist instead).
- CONDITION(I) always returns SIGNAL.

External Programming Interface

NetMaster REXX, like TSO/E REXX, supports an external API.

More information:

REXX External Assembler API (see page 83)

Command Descriptions

This section describes the NetMaster REXX-related commands that you can issue from your region. You can issue these commands to execute REXX, or control or manage the REXX execution environment.

Get Help About a Command

Your region provides online help on commands.

To get help about a command, enter **HELP** followed by the command at the command prompt.

The online help for the command appears.

Note: You can also type the command and press F1 to get help.

Example: Get Help About the SHOW REXX Command from OCS

At the command prompt, type **HELP SHOW REXX**.

SHOW

The SHOW command supports the following operands:

REXX [ALL | USER=*userid* | ID=*n*]

Displays a list of executing REXX processes. If no additional operands are entered, all REXX executing for the current user is assumed. The ALL operand shows all executing REXX processes. A specific user ID can be nominated, or a specific NCLID (REXXID) can be used to obtain information about a specific REXX process.

REXXSTAT [=*pattern*] [STATS] [DETAILS]

Displays the status of loaded REXX procedures. It includes information and statistics related to loading and compiling procedures.

The list of procedures can be filtered using a pattern, and additional statistics are displayed if the STATS option is specified. If you specify DETAILS, information is displayed about each in-storage procedure, including storage usage and compilation and load times.

FLUSH

Use the FLUSH command to flush a currently executing NCL or REXX process.

Note: For more information, see the online help.

GO

Use the GO command to provide input data to a REXX process that is waiting on input.

Note: For more information, see the online help.

LOAD

Use the LOAD command to load REXX procedures into storage for later execution. Use the LIB option if you want to load from a test library rather than from the COMMANDS concatenation.

Notes:

- For more information, see the online help.
- Preloading procedures that are in use causes the procedures to be marked as pending-unload. The current users are allowed to keep using the in-storage copy. New users are directed to a freshly-loaded copy. When all current users are finished, the old copy is purged.

START

Use the START command to start a REXX process provided that SYSPARMS AUTOREXX=YES is in effect.

Notes:

- For more information, see the online help.
- If SYSPARMS AUTOEXEC=YES is in effect, and if the command does not start with a recognizable command name, but the first word is a valid PDS member name, then the command is treated as an implied START command.

UNLOAD

Use the UNLOAD command to remove the existing version of a procedure from storage so that a new version can be loaded (for example, after changing the source on disk).

Note: For more information, see the online help.

REXX

Use the REXX command to explicitly request that a REXX procedure be executed.

This command has the following format:

```
REXX procname [ arguments… ]
```

procname

Specifies the name of the procedure to execute. The procedure name must be a valid PDS member name.

arguments

Specifies argument string to the REXX procedure.

When a REXX process terminates, a message is issued, indicating the result (return code).

RXCHECK

Use the RXCHECK command to compile a syntax-check or to load a REXX program. The command performs the compilation (or load if object) and reports on the results. However, the generated object code is not retained.

This command has the following format:

```
RXCHECK procname
   [ LIBRARY=libname ]
   [ LIST | NOLIST ]
   [ STOP | NOSTOP ]
   [ WARN | NOWARN ]
   [ STATS | NOSTATS]
   [ QSFIX | NOQSFIX]
```

procname

Specifies the name of the REXX program to check.

LIBRARY=*libname*

Specifies the library in which the program is located. For z/OS, this is the ddname of the PDS concatenation. For VM, it is the file type. If omitted, the standard library is used.

LIST | NOLIST

Controls the production of a compile listing. Specifying LIST provides a source listing.

Default: NOLIST

STOP | NOSTOP

Controls whether the compiler attempts to continue after encountering an error. By default, compilation stops on the first error. Specifying NOSTOP causes the compiler to attempt to continue to locate additional errors. This may not be successful. Some errors may cause additional spurious errors as the compiler attempts to recover, and sometimes an error is fatal.

Default: STOP

WARN | NOWARN

Controls whether warnings are reported.

Default: WARN

STATS | NOSTATS

Controls the production of compilation statistics at the end.

Default: NOSTATS

QSFIX | NOQSFIX

Defaults to the value of the setting of system parameter RXQSFIX. However, if set, this operand overrides this sysparm setting. Use this operand during REXX checking.

During RXCHECK processing, if this operand is set to QSFIX, any REXX source containing a quoted string that extends across lines and is less than 250 bytes in length after trailing blanks have been stripped, is converted. A warning message is displayed when this processing occurs if WARN (the default) is specified.

RXCTL

The RXCTL command can be used to do the following:

- Control currently executing REXX processes.
- Compile a REXX source procedure and write object out.

Note: For more information, see the online help.

RXCTL OFFLOAD

Use the RXCTL OFFLOAD command to save a compiled version of a REXX procedure so that it can be loaded without needing compilation.

This command has the following format:

```
RXCTL    OFFLOAD    procname    LIB=ddname
         OUT=ddname
         [QSFIX | NOQSFIX]
```

OFFLOAD

Indicates this is a REXX procedure offload (compile and write object) request.

procname

Specifies the name of the procedure to compile.

LIB=*ddname*

Specifies the input library.

Note: There is no default. (The normal source library is 'COMMANDS').

OUT=*ddname*

Specifies the output library (PDS) to write to. This must be the ddname of an allocation to a single PDS (not a concatenation), with attributes of F(B), LRECL=80. The object is written out to the PDS using the same member name as the input source.

Note: Although the output ddname must be allocated to a single data set, the same data set can be part of a concatenation (under another ddname).

Important! The output library should *not* be the same data set as the source, otherwise the source will be overwritten.

QSFIX | NOQSFIX

Defaults to the value of the setting of system parameter RXQSFIX. However, if set, this operand overrides this sysparm setting. Use this operand during REXX checking.

When this operand is set to QSFIX, any REXX source containing a quoted string that extends across lines and is less than 250 bytes in length after trailing blanks have been stripped is converted. If WARN (the default) is specified, a warning message is displayed when this processing occurs.

Control REXX Processes

You can use the RXCTL command to control currently executing REXX processes. To do this, use the following syntax:

```
[ RXCTL ] { HE | HI | HT | RT | TE | TS }
              [ ALL | ID=id ]
```

Note: The six operands, HE, HI, HT, RT, TE, and TS are also defined as commands. You can use the RXCTL HT or the HT commands; they are the same command.

These operands are analogous to the TSO/E REXX immediate commands. They cause an immediate change to the status of a REXX process.

HE

Halts Execution. This halts execution of a REXX process.

HI

Halts Interpretation. This halts interpretation of a REXX process.

HT

Halts Typing. This halts typing and causes all output (SAY and TRACE) to be discarded.

RT

Resumes Typing. This resumes typing and causes all output (SAY and TRACE) to be resumed.

Note: TE and TS are recognized but not supported.

If none of the operands listed previously is specified, there must be one active REXX process in the current environment (window, INTCMD environment). That process is affected as requested. If there is more than one process, a message is issued.

If the ALL operand is specified, the RXCTL command affects all REXX processes executing in the current environment.

If the ID=*n* operand is specified, the specified REXX process is affected.

Chapter 2: ADDRESS Environments

More information:

NetMaster REXX Environment (see page 39)
Tivoli NetView Emulation (see page 63)
NCCF Commands (see page 97)
REXX Analyzer (see page 67)

This section contains the following topics:

About ADDRESS Environments

NetMaster REXX supports the following ADDRESS environments:

NM

Specifies the CA NetMaster facility environment.

Note: The NM environment also supports the standard commands listed in Standard Commands (see page 34).

NETVIEW

Specifies the Tivoli NetView emulation environment.

Note: The Tivoli NetView environment also supports the standard commands listed in Standard Commands (see page 34).

NETVASIS

Specifies another Tivoli NetView emulation environment.

Note: The NETVASIS environment also supports the standard commands listed in Standard Commands (see page 34).

MVS

Specifies an environment where several base commands are available.

LINK

Links a program.

ATTACH

Attaches a program.

SYSVIEWE

Specifies the CA SYSVIEW interface environment.

CA-7

Specifies the CA 7 WA product interface environment.

CASCHD

Is the CA Scheduler JM product interface environment.

Standard Commands

Some environments supported by NetMaster REXX support the following set of standard commands.

DELSTACK

Deletes the most recently created stack.

DROPBUF

Drops the most recently created stack buffer.

EXECIO (see page 42)

Performs data set I/O.

HT

Halts typing.

MAKEBUF

Makes a buffer.

NEWSTACK

Makes a stack.

QBUF

Queries buffers.

QELEM

Queries elements.

QSTACK

Queries stacks.

RT

Resumes typing.

SLEEP

Waits for a period of time.

SUBCOM

Queries a command environment.

TE

Ends tracing (recognized but not supported).

TS

Starts tracing (recognized but not supported).

Note: For more information about these commands (except for SLEEP), see the *TSO/E REXX Reference*.

SLEEP REXX Command—Wait for a Specified Period

The SLEEP command suspends the REXX process until the specified period elapses.

This command has the following format:

```
SLEEP nnnnn[.nn]
```

***nnnnn*[.*nn*]**

Specifies the period in seconds for which the REXX process is suspended.

Range: 0-86400 (24 hours)

Example: You want to suspend the process for half a second and specify the following:

```
SLEEP 0.5
```

The command can return *one* of the following codes:

0

Indicates that the waiting period has elapsed

-200

Indicates invalid specification.

ADDRESS Environment Descriptions

The following describes the ADDRESS environments supported by NetMaster REXX.

Note: For more information about these environments, see the *TSO/E REXX Reference*.

MVS

In addition to the standard commands (see page 34), MVS supports EX or EXEC and implied EXEC.

LINK and ATTACH

These environments can be used to call user-written programs. They are supported as documented in the *TSO/E REXX Reference*.

Note: The LINKPGM, LINKMVS, ATTCHPGM, and ATTCHMVS ADDRESS environments are recognized, but not presently supported. Commands referencing them always return code 3.

CA Product Interfaces

The following sections describe the CA product interface ADDRESS environments that are available to NetMaster REXX provided that the product is installed and the interface module (GSVXAPIE, CAL2X2WR, or CAJCADDR) is available in the STEPLIB or JOBLIB concatenation, or through LINKLIST or LPALIST. The availability is tested during initialization only. If you copy the module into the load library after initialization, the interface is not available until you stop and restart the product region.

SYSVIEWE

This ADDRESS environment provides an interface to CA SYSVIEW if it is installed. The interface module is GSVXAPIE.

Note: For more information, see the CA SYSVIEW documentation.

CA-7

This ADDRESS environment provides an interface to CA 7 WA if it is installed. The interface module is CAL2X2WR.

Note: For more information, see the CA 7 WA documentation.

CASCHD

This ADDRESS environment provides an interface to CA Scheduler JM if it is installed. The interface module is CAJCADDR.

Note: For more information, see the CA Scheduler JM documentation.

Chapter 3: NetMaster REXX Environment

This chapter describes the commands in the subcommand environment that enable REXX programs to interact with CA NetMaster.

This section contains the following topics:

CALL

The CALL command enables REXX to call NCL procedures, including parameters and shared variables. You must explicitly name any variables that you want to share. *Do not use prefixes.*

This command has the following format:

```
CALL PROC=procname [PARMm=value1,..valuem] [SHAREn=name1,..namen]
```

PROC=*procname*

Specifies the PDS member name that contains a valid NCL procedure.

PARM*n*=*value*

Specifies a valid REXX variable name.

Range: 1-20

SHARE*n*=*name*

Specifies the explicit shared variable name (no ampersand). You can specify only variables that are valid names in NCL and REXX.

Range: 1-50

Restrictions

- If the target procedure is NCL, each parameter and input-shared variable's value is limited to 256 characters (NCL restriction).
- The name of each shared variable must be a valid NCL variable name.

Example: CALL Command

```
'CALL PROC=$esapi00 PARM1=CLASS=translate PARM2=data=AHRSP',
     'SHARE1=$ESXLATEFullTxt SHARE2=$ESXLATEShortTxt'
say $ESXLATEFullTxt
say $ESXLATEShortTxt
```

CMD/COMMAND

The CMD or COMMAND command enables a REXX procedure to issue a CA NetMaster command.

This command has the following format:

```
CMD command_text
```

or

```
COMMAND command_text
```

command_text

Specifies any valid REXX text, variable, or expression containing the command.

After checking that the required *command_text* is specified, the *command_text* is processed. No validity checking of *command_text* is performed. No command execution result is returned, and RC=0 is always returned.

Note: Unlike all other REXX subcommand environments, the CA NetMaster command always executes asynchronously. Consequently, neither a command completion message nor a return code is provided. However, you can use the ADDRESS NETVIEW PIPE NM command to execute these commands and process their responses.

Example: CMD/COMMAND

```
trace o
ADDRESS 'NM'
'cmd syscmd d t'
sysc = 'syscmd d a,l'
cmd sysc
```

EXECIO

NetMaster REXX uses the EXECIO command to read and write data to or from a file. You can use EXECIO to do the following:

- Read from a data set to the REXX data stack for serialized processing, or to a list of variables for random processing.
- Write data to a data set from the REXX data stack or a list of variables.

EXECIO emulates the TSO/E EXECIO command with the following exceptions:

DISKRU

DISKRU is not supported.

DISKW (empty stack for non 3270 session)

If an EXECIO * DISKW command is issued from a non-3270 type session and the REXX data stack becomes empty before a null record is found (which would normally terminate EXECIO), then the EXECIO request completes normally instead of requesting input from an arbitrary input data stream (because there is no equivalent for SYSTSIN DD in CA NetMaster).

Return Code 4 is not used

TSO/E EXECIO terminates with a return code of 4 when an empty data set is found within a concatenation list during a DISKR operation. NetMaster REXX EXECIO does not have that restriction.

Note: for more information about EXECIO, see your IBM TSO/E documentation.

GLOBALV

The GLOBALV command can be used to set and retrieve the value of an NCL global variable. These variables are also visible to NCL procedures in the same region.

This command has the following format:

```
GLOBALV GET vname|vnmlist
GLOBALV PUT vname|vnmlist
```

vname

Specifies the variable name. The name must be prefixed with GLBL.

Characters: A-Z,a-z,0-9, $, #, @

Range: The global variable name *vname* must not be longer than 12 characters. The variable name is not case sensitive. Do not include the ampersand (&) character.

vnmlist

Specifies a list of variable names, for example, *vname*[, …*vname*].

GET

Retrieves the value of the given NCL global variable and assigns it to the identically named REXX variable.

PUT

Takes the value of the REXX variable and uses it to set the value of the identical NCL global variable.

If the input variable is not specified, a null value is assigned to the output global variable. Conversely, the variable name is assigned to the REXX output variable, per the REXX standard.

Example: GLOBALV

```
globalv get 'GLBL$RMOSVER'
GLBL$kb3a = GLBL$RMOSVER
globalv put 'GLBL$kb3a'
globalv get 'GLBL$kb3a'
say 'rexxvx=<'||GLBL$kb3a||'>'
```

VARTABLE

The VARTABLE command allows a REXX procedure to perform most functions of the VARTABLE NCL verb.

A *vartable* is a table of variables.

This command can execute *one* of the following operations:

- ADD
- ALLOC
- DELETE
- FREE
- GET
- PUT
- QUERY
- RESET
- UPDATE

Notes:

- For information about the VARTABLE NCL verb, see the *Network Control Language Reference*.
- For all VARTABLE actions, the result is returned in RC. The result is equal to &ZFDBK returned by the execution of the &VARTABLE NCL verb.

VARTABLE ADD

VARTABLE ADD adds an entry to an existing memory-resident vartable.

This command has the following format:

```
VARTABLE ADD ID=tablename [SCOPE=scope] KEY=kname
[ADJUST=a|COUNTER=c]
[FIELDS=(fldlist)] [VARS=(varlist)]
```

ID=*tablename*

Specifies the name of the vartable.

Characters: A–Z, a–z, 0–9, #, $, and @

Limits: 1–12 characters

SCOPE = {SYSTEM|AOM|GLOBAL|PROCESS|REGION}

Specifies the scope of the vartable.

Default: PROCESS

KEY=*kname*

Specifies the name of the variable that contains the value of the key to assign to this table entry.

ADJUST=*a*

Specifies the value by which to increase or decrease the counter.

COUNTER=*c*

Specifies the value of the new or updated entry.

FIELDS=*fldlist*

Specifies the fields to add in the format *fname*[,…*fname*].

VARS=*varlist*

Specifies a list of valid REXX variables that holds values assigned to *fldlist*. If *fldlist* includes DATA*, the associated VARS list entry must be in the format *prefix**. When values are returned, they are in variables *prefix*1 through *prefixn*.

Example: VARTABLE ADD

```
dt11 = 'kbrxd11'
dt12 = 'kbrxd12'
dt13 = 'kbrxd13'
'vartable add id=rxtest key=secnd scope=region ',
'fields=(data1,data2,data3) vars=(dt11,dt12,dt13)'
say 'kbrexx06_add=<'||RC||'>'
```

Return Codes

The return codes are as follows:

0

Indicates that the request was satisfied.

1

Indicates that the entry was added successfully. The table was at the limit specified by the VARTABLE ALLOC, and DELOLD=YES was specified with VARTABLE ALLOC. The oldest entry was deleted to make room for this entry.

4

Indicates that an entry with that key value already exists.

12

Indicates that the supplied key value was longer than the table key length.

16

Indicates that no table of this name exists in this scope.

24

Indicates that the table is already at the limit specified. The entry could not be added.

28

Indicates that the variable specified for AOMID is longer than 12 characters.

32

Indicates that SCOPE=AOM table has been disabled due to a storage error.

100

Indicates that the variable specified for AOMATTR is longer than 30 characters.

101 to 130

Indicates that variable specified AOMATTR has an invalid value.

VARTABLE ALLOC

VARTABLE ALLOC defines a new memory-resident vartable. The table, as defined, contains no entries. After this table is defined, other VARTABLE operations can refer to the table.

Specifying USERCORR indicates that the user wants to have synchronized correlation protection on entries in the vartable. This operand allows control over the use of the USERCORR field in table entries when performing VARTABLE UPDATE, VARTABLE DELETE, or VARTABLE PUT operations.

This command has the following format:

```
VARTABLE ALLOC ID=tablename [SCOPE=scope] [KEYLEN=klen]
[AGE={NO|NEW|ALL|UPDATE|GET}] [DELOLD={NO|YES}]
[KEYFMT=fmt] [DATA=dn] [LIMIT=ln] [USERCORR=NO|YES]
```

ID=*tablename*

Specifies the name of the vartable.

Characters: A–Z, a–z, 0–9, #, $, and @

Limits: 1–12 characters

SCOPE = {SYSTEM|AOM|GLOBAL|PROCESS|REGION}

Specifies the scope of the vartable.

Default: PROCESS

KEYLEN=*klen*

Specifies the length of the keys in this table. Required if *keyfmt*=CHAR is specified or assumed.

Range: 1-256

AGE={NO|NEW|ALL|UPDATE|GET}

Specifies whether entries are aged when certain operations are performed on them.

Default: NO

DELOLD={NO|YES}

Specifies whether VARTABLE ADD and VARTABLE PUT can delete the oldest entry automatically when the table is full.

Default: NO

KEYFMT={CHAR | NUM}

Specifies whether the table has a numeric or character format.

Default: CHAR

DATA=*dn*

Specifies how many data fields can be stored in each table entry.

Range: 1-16

Default: 1

LIMIT=*ln*

Specifies whether the table has a limit on the number of entries.

Range: 0-1,000,000

Default: 0 indicates that the table can have any number of entries.

USERCORR={NO|YES}

Specifies whether the USERCORR field can be used in table entries when performing VARTABLE UPDATE and VARTABLE PUT operations.

Default: NO

Example: VARTABLE ALLOC

```
'vartable alloc id=rxtest scope=region keylen=11 data=4 limit=5'
```

Return Codes

The codes returned are the same as the &ZFDBK values as documented for the NCL &VARTABLE verb.

The return codes are as follows:

0

Indicates that the request was satisfied.

16

Indicates that no table of this name exists in this scope.

20

Indicates that the maximum number of tables has already been allocated.

24

Indicates that the allocation exceeds the storage allocated for mirroring.

VARTABLE DELETE

VARTABLE DELETE deletes an entry from an existing memory-resident vartable.

This command has the following format:

```
VARTABLE DELETE ID=tablename [SCOPE=scope] [KEY=kname]
[FIELDS=(fldlist)VARS=(varlist)]
```

ID=*tablename*

Specifies the name of the vartable.

Characters: A–Z, a–z, 0–9, #, $, and @

Limits: 1–12 characters

SCOPE = {SYSTEM|AOM|GLOBAL|PROCESS|REGION}

Specifies the scope of the vartable.

Default: PROCESS

KEY=*kname*

Specifies the name of the variable that contains the value of the key that identifies this table entry.

FIELDS=*fldlist*

Specifies the fields to delete in the format *fname*[,...*fname*].

VARS=*varlist*

Specifies a list of valid REXX variables that holds values assigned to *fldlist*. If *fldlist* includes DATA*, the associated VARS list entry must be in the format *prefix**. When values are returned, they are in variables *prefix*1 through *prefixn*.

Example: VARTABLE DELETE

```
KEY='KEY001'
'VARTABLE DELETE ID=MYTABLE KEY='||KEY
```

Return Codes

The codes returned are the same as the &ZFDBK values as documented for the NCL &VARTABLE verb.

The return codes are as follows:

0

Indicates that the request was satisfied.

4

Indicates that no entry with the requested key value exists.

8

Indicates that an entry with the supplied key value already exists, and the supplied user correlator value did not match the user correlator value in that entry.

12

Indicates that the supplied key value was longer than the table key length.

16

Indicates that no table of this name exists in this scope.

32

Indicates that SCOPE=AOM table has been disabled due to a storage error.

VARTABLE FREE

VARTABLE FREE deletes (frees) an existing memory-resident vartable. Use it to delete all entries in the table and the table definition itself. It also frees all storage associated with the table.

This command has the following format:

```
VARTABLE FREE ID=tablename [SCOPE=scope]
```

ID=*tablename*

Specifies the name of the vartable.

Characters: A–Z, a–z, 0–9, #, $, and @

Limits: 1–12 characters

SCOPE = {SYSTEM|AOM|GLOBAL|PROCESS|REGION}

Specifies the scope of the vartable.

Default: PROCESS

Example: VARTABLE FREE

```
'vartable free  id=rxtest scope=region'
```

Return Codes

The codes returned are the same as the &ZFDBK values as documented for the NCL &VARTABLE verb.

The return codes are as follows:

0

Indicates that the request was satisfied.

16

Indicates that no table of this name exists in this scope.

20

SCOPE=AOM table was in use at the same time that you attempted to delete the mirrored copy. The table remains allocated. We recommend that you retry the operation after one second.

VARTABLE GET

VARTABLE GET retrieves (gets) an entry from an existing memory-resident vartable. It is not necessary to know the exact key of the record.

This command has the following format:

```
VARTABLE GET ID=tablename [SCOPE=scope] [KEY=kname]
[AGE={NO|YES}] [DELETE={NO|YES}]
[FIELDS=(fldlist) VARS=(varlist)]
```

ID=*tablename*

Specifies the name of the vartable.

Characters: A–Z, a–z, 0–9, #, $, and @

Limits: 1–12 characters

SCOPE = {SYSTEM|AOM|GLOBAL|PROCESS|REGION}

Specifies the scope of the vartable.

Default: PROCESS

KEY=*kname*

Specifies the name of the variable that contains the value of the key that identifies this table entry.

AGE={NO|YES}

Specifies whether to make the retrieved entry the newest in the table.

DELETE={NO|YES}

Specifies whether to delete the retrieved entry.

Default: NO

FIELDS=*fldlist*

Specifies the fields to get in the format *fname*[,...*fname*].

VARS=*varlist*

Specifies a list of valid REXX variables that holds values assigned to *fldlist*. If *fldlist* includes DATA*, the associated VARS list entry must be in the format *prefix**. When values are returned, they are in variables *prefix*1 through *prefixn*.

Example: VARTABLE GET

```
/* get the stem of IP connections from $IPLINK vartable              */
lk.0 = 0
lk.1 = 'a'
'vartable query id=$IPLINKS scope=system fields=(total) vars=(lk.0)'
If RC = 0 Then Do i = 1 to lk.0
  'vartable get id=$IPLINKS scope=system fields=(key) vars=(lk.i) ',
  'opt=KGT key="'lk.i'"'
End
```

Return Codes

The codes returned are the same as the &ZFDBK values as documented for the NCL &VARTABLE verb.

The return codes are as follows:

0

Indicates that the request was satisfied.

4

Indicates that no entry with the requested key value exists.

12

Indicates that the supplied key value was longer than the table key length.

16

Indicates that no table of this name exists in this scope.

32

Indicates that SCOPE=AOM table has been disabled due to a storage error.

VARTABLE PUT

VARTABLE PUT adds to or updates an entry in an existing memory-resident vartable. If there is no entry with a matching key, the entry is added. If an entry with a matching key already exists, the entry is updated.

The PUT operation can occur concurrently with vartable updating in other processes. If you want to delete or update a vartable entry, you should use a correlator to synchronize any accesses to the entry.

This command has the following format:

```
VARTABLE PUT ID=tablename KEY=kname [SCOPE=scope]
[ADJUST=a|COUNTER=c]
[FIELDS=(fldlist) VARS=(varlist)]
```

ID=*tablename*

Specifies the name of the vartable.

Characters: A–Z, a–z, 0–9, #, $, and @

Limits: 1–12 characters

SCOPE = {SYSTEM|AOM|GLOBAL|PROCESS|REGION}

Specifies the scope of the vartable.

Default: PROCESS

KEY=*kname*

Specifies the name of the variable that contains the value of the key that identifies this table entry.

ADJUST=*a*

Specifies the value by which to increase or decrease the counter.

COUNTER=*c*

Specifies the value of the new or updated entry.

FIELDS=*fldlist*

Specifies the fields to put in the format *fname*[,...*fname*].

VARS=*varlist*

Specifies a list of valid REXX variables that holds values assigned to *fldlist*. If *fldlist* includes DATA*, the associated VARS list entry must be in the format *prefix**. When values are returned, they are in variables *prefix*1 through *prefixn*.

Example: VARTABLE PUT

```
'vartable put id=rxtest key=myrexx scope=region ',
'fields=(data4) vars=(dta4)'
```

More information:

VARTABLE GET (see page 52)

Return Codes

The codes returned are the same as the &ZFDBK values as documented for the NCL &VARTABLE verb.

The return codes are as follows:

0

Indicates that the request was satisfied.

1

Indicates that the entry was added successfully. The table was at the limit specified by the VARTABLE ALLOC, and DELOLD=YES was specified with VARTABLE ALLOC. The oldest entry was deleted to make room for this entry.

8

Indicates that an entry with the supplied key value already exists, and the supplied user correlator value did not match the user correlator value in that entry.

12

Indicates that the supplied key value was longer than the table key length.

16

Indicates that no table of this name exists in this scope.

20

Indicates that the table was allocated with USERCORR=YES specified, and no user correlator was supplied.

28

Indicates that the variable specified for AOMID is longer than 12 characters.

32

Indicates that SCOPE=AOM table has been disabled due to a storage error.

100

Indicates that the variable specified for AOMATTR is longer than 30 characters.

101 to 130

Indicates that variable specified AOMATTR has an invalid value.

VARTABLE QUERY

VARTABLE QUERY obtains information about a memory-resident vartable. You use it to inquire about the existence of a given vartable. If it exists, you can optionally retrieve attribute information.

This command has the following format:

```
VARTABLE QUERY ID=tablename [SCOPE=scope]
[FIELDS=(fldlist) VARS=(varlist)]
```

ID=*tablename*

Specifies the name of the vartable.

Characters: A–Z, a–z, 0–9, #, $, and @

Limits: 1–12 characters

SCOPE = {SYSTEM|AOM|GLOBAL|PROCESS|REGION}

Specifies the scope of the vartable.

Default: PROCESS

FIELDS=*fldlist*

Specifies the information that you want returned in the format *fname*[,...*fname*].

Valid Values: keylen, age, delold, data, limit, usercorr, total

VARS=*varlist*

Specifies a list of valid REXX variables that holds values assigned to *fldlist*. If *fldlist* includes DATA*, the associated VARS list entry must be in the format *prefix**. When values are returned, they are in variables *prefix*1 through *prefixn*.

Example: VARTABLE QUERY

```
'vartable query id=rxtest scope=region ',
'fields=(keylen,limit,delold) vars=(klen,lmt,dlo)'
say 'kbrexx06_klen=<'||klen||'><'||lmt||'><'||dlo||'>'
```

Return Codes

The codes returned are the same as the &ZFDBK values as documented for the NCL &VARTABLE verb.

The return codes are as follows:

0

Indicates that the request was satisfied.

16

Indicates that no table of this name exists in this scope.

VARTABLE RESET

VARTABLE RESET deletes all entries from an existing memory-resident vartable. It lets you delete multiple entries from an existing vartable while preserving its definition.

This command has the following format:

```
VARTABLE RESET ID=tablename [SCOPE=scope]
[OLDEST=n | NEWEST=n]
```

ID=*tablename*

Specifies the name of the vartable.

Characters: A–Z, a–z, 0–9, #, $, and @

Limits: 1–12 characters

SCOPE = {SYSTEM|AOM|GLOBAL|PROCESS|REGION}

Specifies the scope of the vartable.

Default: PROCESS

OLDEST=*n*

Specifies the number of oldest entries to delete.

NEWEST=*n*

Specifies the number of newest entries to delete.

Example: VARTABLE RESET

```
'vartable reset id=rxtest scope=region'
```

Return Codes

The codes returned are the same as the &ZFDBK values as documented for the NCL &VARTABLE verb.

The return codes are as follows:

0

Indicates that the request was satisfied.

16

Indicates that no table of this name exists in this scope.

20

SCOPE=AOM table in use at the same time that you attempted to delete the mirrored copy. The table remains allocated. We recommend that you retry the operation after one second.

VARTABLE UPDATE

VARTABLE UPDATE updates an entry in an existing memory-resident vartable. The update operation can occur concurrently with vartable updating in other processes. If you want to delete or update a vartable entry, you should use a correlator to synchronize any accesses to the entry.

This command has the following format:

```
VARTABLE UPDATE ID=tablename [SCOPE=scope] KEY=kname
[ADJUST=a|COUNTER=c]
[FIELDS=(fldlist) VARS=(varlist)]
```

ID=*tablename*

Specifies the name of the vartable.

Characters: A–Z, a–z, 0–9, #, $, and @

Limits: 1–12 characters

SCOPE = {SYSTEM|AOM|GLOBAL|PROCESS|REGION}

Specifies the scope of the vartable.

Default: PROCESS

KEY=*kname*

Specifies the name of the variable that contains the value of the key that identifies this table entry.

ADJUST=*a*

Specifies the value by which to increase or decrease the counter.

COUNTER=*c*

Specifies the value of the new or updated entry.

FIELDS=*fldlist*

Specifies the fields to update in the format *fname*[,...*fname*].

VARS=*varlist*

Specifies a list of valid REXX variables that holds values assigned to *fldlist*. If *fldlist* includes DATA*, the associated VARS list entry must be in the format *prefix**. When values are returned, they are in variables *prefix*1 through *prefixn*.

Example: VARTABLE UPDATE

```
'vartable update id=rxtest key=secnd scope=region ',
'fields=(data1,data2,data3) vars=(dt11,dt12,dt13) ',
'adjust=15'
```

Return Codes

The codes returned are the same as the &ZFDBK values as documented for the NCL &VARTABLE verb.

The return codes are as follows:

0

Indicates that the request was satisfied.

4

Indicates that no entry with the requested key value exists.

8

Indicates that an entry with the supplied key value already exists, and the supplied user correlator value did not match the user correlator value in that entry.

12

Indicates that the supplied key value was longer than the table key length.

16

Indicates that no table of this name exists in this scope.

20

Indicates that the table was allocated with USERCORR=YES specified, and no user correlator was supplied.

28

Indicates that the variable specified for AOMID is longer than 12 characters.

32

Indicates that SCOPE=AOM table has been disabled due to a storage error.

100

Indicates that the variable specified for AOMATTR is longer than 30 characters.

101 to 130

Indicates that variable specified AOMATTR has an invalid value.

WRITE

The WRITE command lets you issue messages to various destinations. It has more flexibility than the standard REXX SAY statement.

This command has the following format:

```
WRITE
   [LOG={NO|YES}]
   [TERM={NO|YES}]
   [MON={NO|YES}]
   [{COLOR|COLOUR}=color]
   [{HLIGHT|HLITE}=hlight]
   [ALARM={NO|YES}]
   [INTENS={LOW|HIGH}]
   DATA="wtext"
```

LOG={NO | YES}

Specifies whether to write the message to the activity log.

Default: NO

TERM={NO | YES}

Specifies whether to write the message to the owning environment.

Default: YES

MON={NO | YES}

Specifies whether to write the message to all OCS users and dependent environments profiled to receive Monitor class messages.

Default: YES

COLOR|COLOUR={BLUE|GREEN|PINK|RED|TURQUOISE|WHITE|YELLOW|NONE}

Specifies the color of the message.

HLIGHT|HLITE={BLINK|REVERSE|USCORE|NONE}

Specifies the extended highlighting of the message.

ALARM={NO | YES}

Specifies whether to ring the terminal alarm when displayed on an OCS window.

Default: NO

INTENS={NORMAL | HIGH}

Specifies whether to display the message in high or normal intensity.

Default: NORMAL

DATA="*wtext*"

Specifies the text to display on the terminal.

Example: WRITE

```
ADDRESS 'NM'
'WRITE LOG=YES DATA="a message from REXX to the log"'
'WRITE LOG=YES TERM=YES COLOR=RED DATA="Hello World from NM rexx!"'
'WRITE LOG=NO TERM=YES COLOR=GREEN HLITE=REVERSE INTENS=HIGH',
'DATA="This should be green, intensive and reverse!"'
attr=' COLOR=PINK HLITE=REVERSE'
line=" DATA='This should be pink, intensive and reverse!'"
'WRITE LOG=NO TERM=YES'attr||line
allarg=' COLOR=GREEN DATA="This should be just green"'
WRITE allarg
```

Return Codes

The return codes are as follows:

0

Indicates that the request was satisfied.

4

Indicates that the target was a closed OCS window.

8

Indicates that the LUNAME specified was not available.

12

Indicates that the USERID or LUNAME specified was not in OCS mode.

16

Indicates that the target was not found in the System Services domain.

24

Indicates that the destination procedure was at queue limit or the storage limit was reached.

28

Indicates that the target for a TYPE=REQ request was not an NCL procedure.

32

Indicates that the contents of the source MDO exceeds the maximum allowable size.

Chapter 4: Tivoli NetView Emulation

More information:

NCCF Commands (see page 97)
REXX Analyzer (see page 67)

This section contains the following topics:

About Tivoli NetView Emulation

NetMaster REXX provides a Tivoli NetView emulation facility that provides an environment to enable a significant number of NetView REXX procedures to run unchanged under CA NetMaster.

This is achieved through the following:

- Support of the NETVIEW and NETVASIS ADDRESS environments.
- Emulation of many NetView REXX commands, including several that are available only to REXX programs.

In addition, an NCCF emulation facility lets you enter NetView REXX commands and procedure names at a user terminal, and have them correctly processed, even if they have the same name as CA NetMaster commands.

To help you move REXX programs from Tivoli NetView, a REXX analyzer facility is provided.

General REXX Execution

In Tivoli NetView, libraries containing REXX source must be allocated to the DSICLD ddname.

In CA NetMaster, the REXX libraries are allocated to the COMMANDS ddname by default, although the actual library name can be overridden at the user level using the NCL Library DDNAME UAMS attribute.

All NetMaster REXX libraries must be F(B), 80-character records.

By default, any REXX procedure executed in CA NetMaster executes with ADDRESS NM as the default address environment. This means that a procedure that expects a Tivoli NetView environment does not work correctly.

To execute a NetView REXX procedure from OCS or command entry, prefix the procedure name with NV. For example, NV STATCHK PU27 executes the STATCHK procedure in a Tivoli NetViewemulation environment.

Note: The NV prefix is not valid in the NCCF emulation facility. All commands entered under this facility are assumed to be Tivoli NetView commands or procedures. If you want to use the CA NetMaster environment from a NetView REXX procedure, you can specify ADDRESS NM. To revert to the Tivoli NetViewenvironment, specify ADDRESS NETVIEW.

Supported Address Environments

NetMaster REXX supports ADDRESS NETVIEW and ADDRESS NETVASIS. These environments enable these programs to execute NetView REXX commands. While NetMaster REXX supports these environments, not all commands are supported.

These address environments support all of the standard commands (such as NEWSTACK and EXECIO).

Note: NetMaster REXX does not support the ADDRESS NETVDATA (data REXX) environment.

More information:

NCCF Commands (see page 97)
ADDRESS Environments (see page 33)

Supported External Functions

NetMaster REXX recognizes all documented NetView REXX external functions. However, not all functions are fully supported.

More information:

REXX External Assembler API (see page 83)

Locate the Tivoli NetView Data Sets Required by Emulation

If you are using a customized command characteristics definition (CCDEF) table or if you need to emulate the VIEW command, the region requires the data sets that contain the Tivoli NetView system definitions and panels.

To locate the Tivoli NetView data sets required by emulation in a region

1. Enter **/PARMS** at the prompt.

   The parameter groups appear.

2. Enter **F NETVEMLDSN** at the Command prompt.

   The cursor locates the parameter group for the data sets.

3. Enter **U** beside the group.

   The Parameter Group panel appears.

4. Specify the data sets required by the region, and then press F6 (Action).

   The specified data sets become known to the region.

5. Press F3 (File).

   The group settings are saved and will be applied each time the region starts.

Chapter 5: REXX Analyzer

This section contains the following topics:

About the REXX Analyzer

The REXX Analyzer lets you analyze and generate reports on your existing REXX libraries. The REXX Analyzer provides information about your existing NetView REXX procedures to help you migrate from Tivoli NetView.

REXX Analyzer Processing

REXX Analyzer processing comprises the following logical processes:

- Analysis process
- Report generation process

Analysis Process

The analysis process analyzes a specified input data set that contains your REXX procedures and writes the output to a member in a preallocated analysis output data set. Each time analysis is run, a new member is created. The report generation process uses the information stored in the member to generate a REXX Analysis report.

You can analyze different REXX procedure data sets. An analysis member is created for each analyzed REXX procedure data set. You can perform different types of analysis on each REXX procedure data set.

You may need to reanalyze a procedure to account for different REXX procedure formats. There may be instances where your REXX procedure is not identified with a /* REXX*/ in the first line of the procedure, or you may be unsure of what procedures you have in your REXX procedure data set. You can use different Scan Types to identify what is, or is not, a REXX procedure. Only those procedures that pass scan processing are analyzed.

Report Generation Process

The report generation process uses an analysis member to build the report.

The analysis member is read during the report generation process, and information stored in this member is used to build the report.

Procedures That Passed Analysis

Some procedures that contain only recognized entities may still require additional work for the following reasons:

- There are differences (see page 13) between the GREXX compiler and the IBM REXX Interpreter.
- Not all parameters are supported.

  Note: The online help in the NCCF-like facility (see page 97) contains more information about the supported parameters.

Using the REXX Analyzer

The REXX Analyzer has a panel-driven user interface that is accessed from your CA NetMaster region.

Note: For more information about criteria fields and report data, see the online help.

Access the Primary Menu

To access the Primary Menu, enter **/REXXAN** at the prompt.

The following panel appears:

```
NMPROD------------------ REXX Analyzer : Primary Menu ------------------/REXXAN
Select Option ===>

   A   - Analyze REXX
   G   - Generate Report
   M   - Maintain Analysis Output
   X   - Exit
```

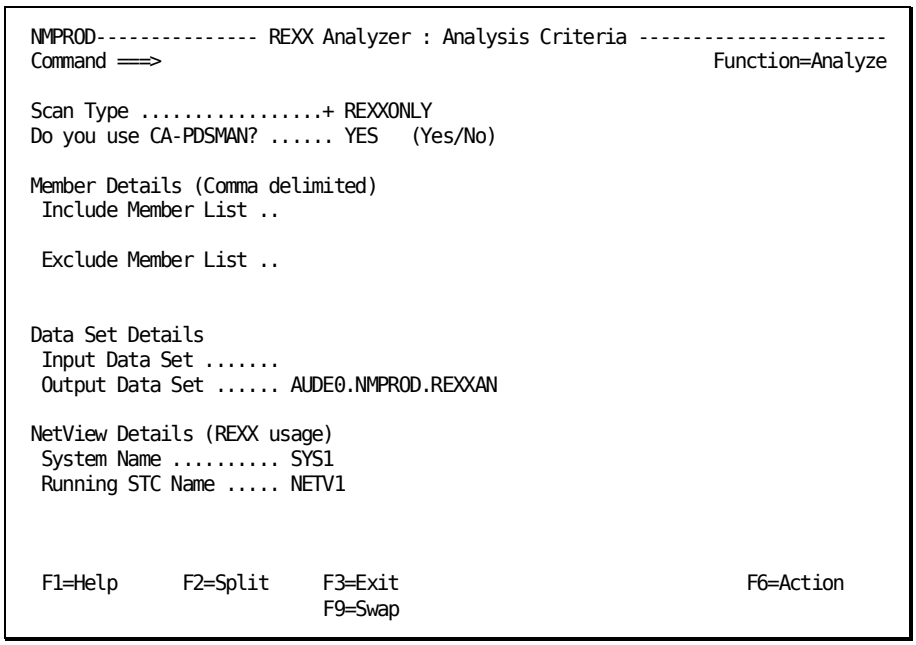
Note: For more information, press F1 (Help).

Analyze a REXX Procedure Library

To analyze a REXX procedure library

1. Enter **A** from the REXX Analyzer Primary Menu.

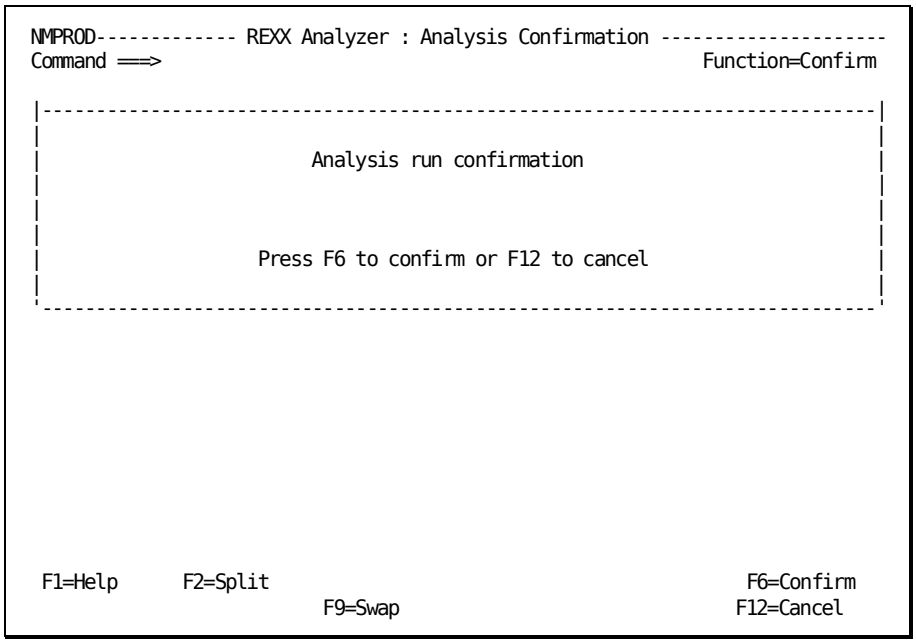
   The following panel appears:



2. Specify the REXX procedure data set name in the Input Data Set field and other criteria as required, and then press F6 (Action) to proceed with analysis processing.

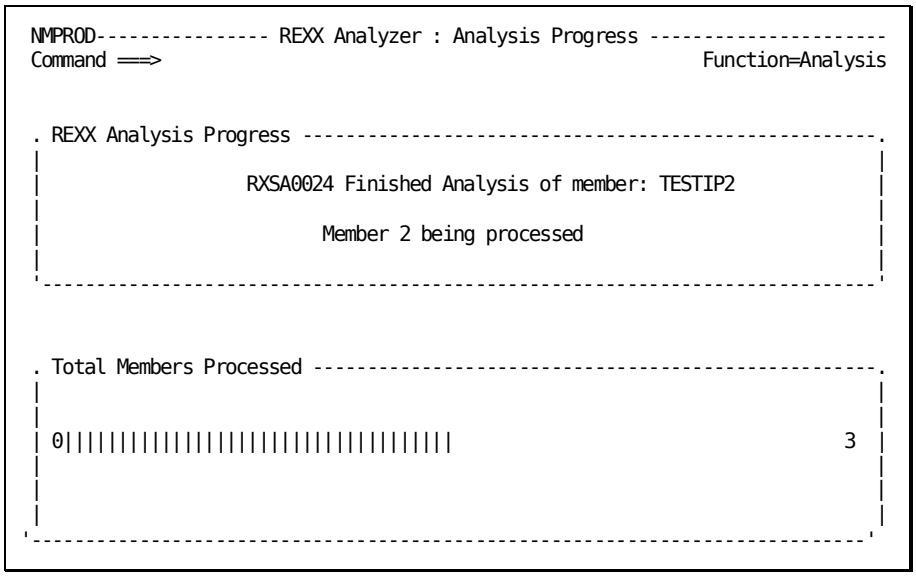
   Note: Ensure your region has read access to the input data set.

   The following panel appears:

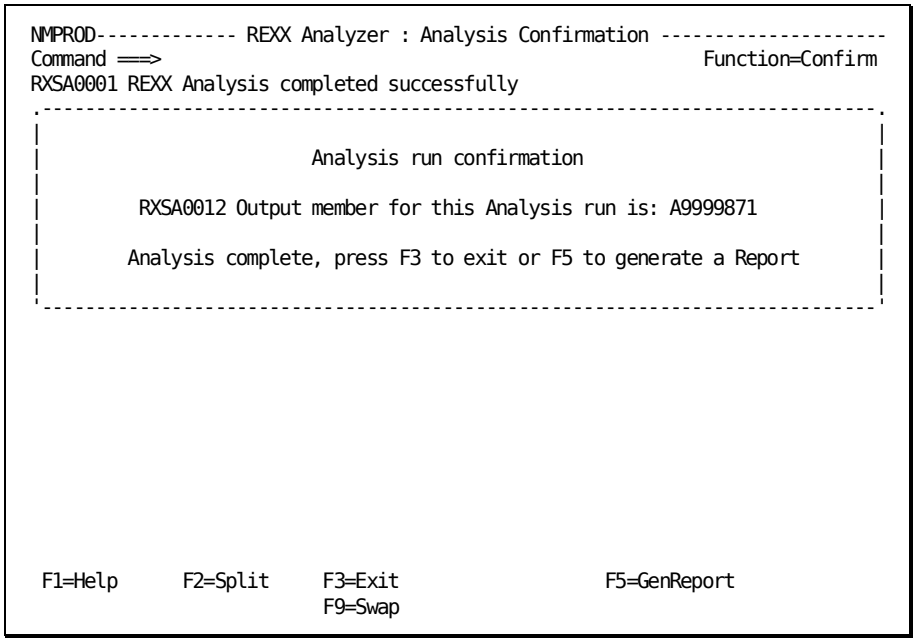


3. Press F6 (Confirm).

   The following panel, showing analysis progress, appears:



   When the analysis is complete, the following panel appears:



4. From the Analysis Confirmation panel, you can do *one* of the following:
   - Press F3 (Exit) to return to the Analysis Criteria panel to run analysis on another REXX procedure data set.
   - Press F5 (GenReport) to produce a report from the generated analysis member.

Generate a Summary Online Report

To generate a summary online report

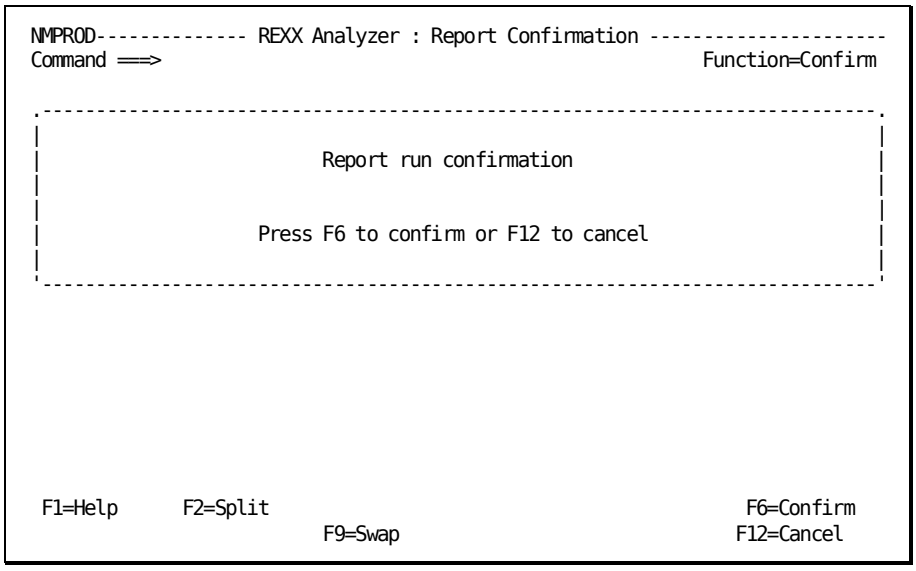
1. Do *one* of the following:
   - Enter **G** from the Primary Menu
   - Press F5 (GenReport) from the Analysis Confirmation panel.

   The following panel appears. The Report Type and Output Type fields specify a summary online report.

```
NMPROD---------------- REXX Analyzer : Report Criteria -----------------------
Command ===>                                                 Function=Report

Report Type .............. SUMMARY_ (Summary or Detailed)

Report ID ................ NETVIEW_

 Output Type ............. ONLINE   (DSN or Online)
 Member Selection .......+ LAST____
 Suppression Character ... ?
 Input Data Set .......... AUDE0.NMPROD.REXXAN_____________________________

Required if Output Type is DSN
 Output Data Set ......... ________________________________________________
 Print Control .........+ NONE
 Line Count .............. 55__




 F1=Help      F2=Split     F3=Exit                              F6=Action
                           F9=Swap
```

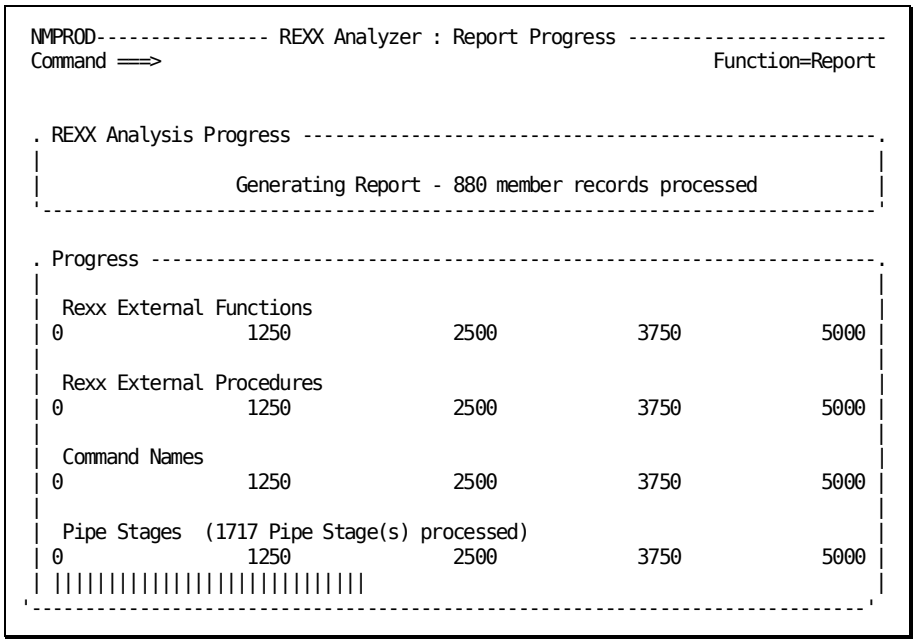
2. Press F6 (Action).

   The following panel appears:

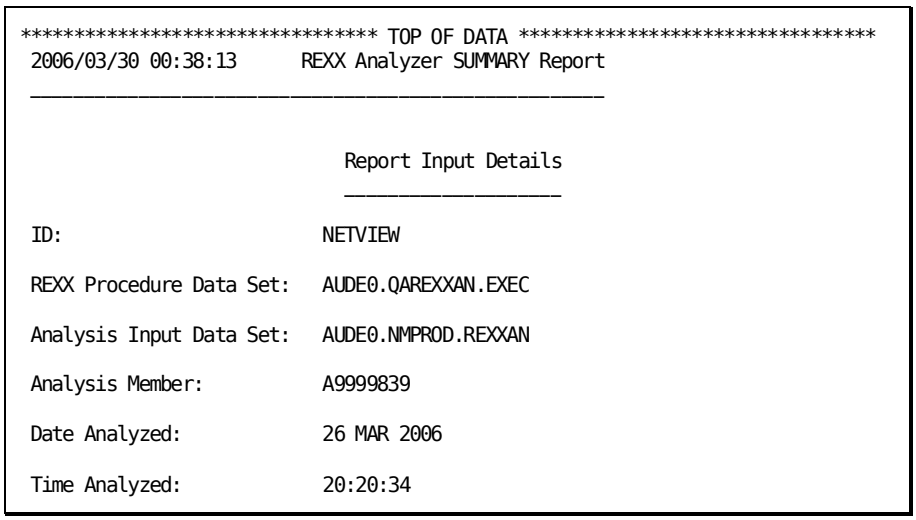


3. Press F6 (Confirm).

   The following panel, showing report generation progress, appears:

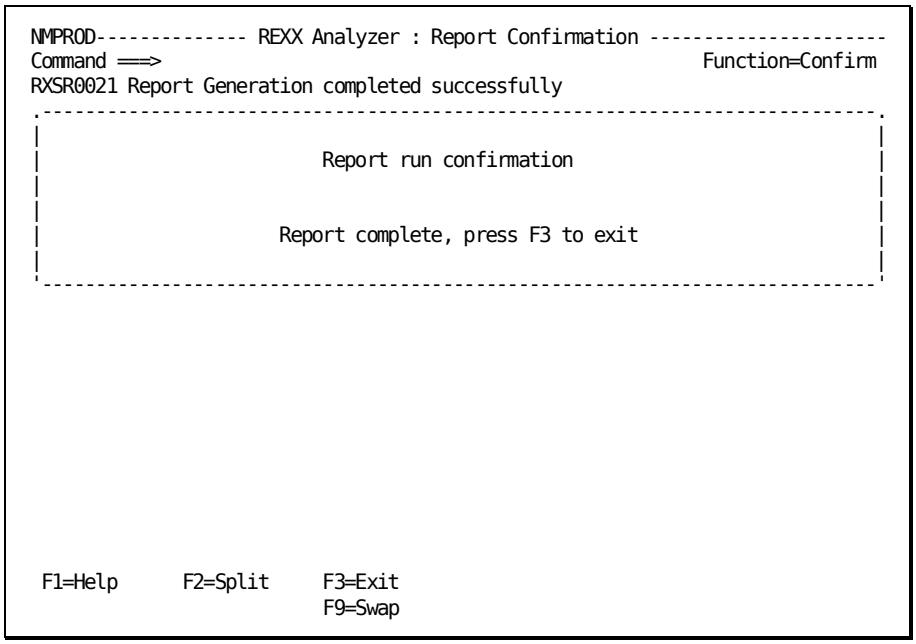


When online report generation is complete, the Online Report panel appears:



4. Press F3 (Exit).

   The following panel appears:



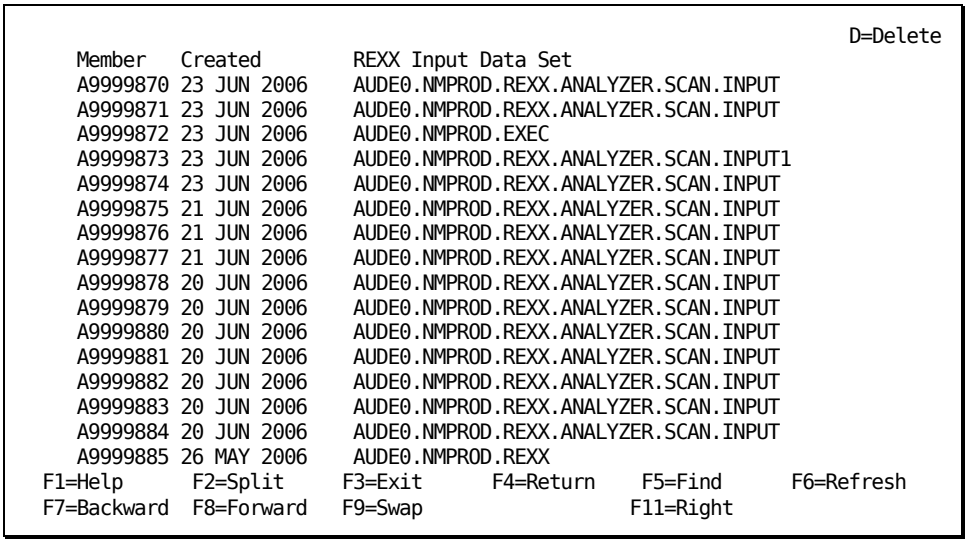
5. From the Report Confirmation panel, you can do *one* of the following:

   - Press F3 (Exit) to return to the Report Criteria panel to generate another report.
   - Press F3 (Exit) twice to return to the Primary Menu.

Maintain Analysis Output

During REXX analysis, an output member is generated. The M - Maintain Analysis Output option on the Primary Menu lets you view existing members and delete any that you no longer require. The listed members are those members that reside in the last used analysis output data set.

When you enter **M** from the Primary Menu, a list similar to the following appears:



```
                                                                  D=Delete
    Member   Created        REXX Input Data Set
    A9999870 23 JUN 2006    AUDE0.NMPROD.REXX.ANALYZER.SCAN.INPUT
    A9999871 23 JUN 2006    AUDE0.NMPROD.REXX.ANALYZER.SCAN.INPUT
    A9999872 23 JUN 2006    AUDE0.NMPROD.EXEC
    A9999873 23 JUN 2006    AUDE0.NMPROD.REXX.ANALYZER.SCAN.INPUT1
    A9999874 23 JUN 2006    AUDE0.NMPROD.REXX.ANALYZER.SCAN.INPUT
    A9999875 21 JUN 2006    AUDE0.NMPROD.REXX.ANALYZER.SCAN.INPUT
    A9999876 21 JUN 2006    AUDE0.NMPROD.REXX.ANALYZER.SCAN.INPUT
    A9999877 21 JUN 2006    AUDE0.NMPROD.REXX.ANALYZER.SCAN.INPUT
    A9999878 20 JUN 2006    AUDE0.NMPROD.REXX.ANALYZER.SCAN.INPUT
    A9999879 20 JUN 2006    AUDE0.NMPROD.REXX.ANALYZER.SCAN.INPUT
    A9999880 20 JUN 2006    AUDE0.NMPROD.REXX.ANALYZER.SCAN.INPUT
    A9999881 20 JUN 2006    AUDE0.NMPROD.REXX.ANALYZER.SCAN.INPUT
    A9999882 20 JUN 2006    AUDE0.NMPROD.REXX.ANALYZER.SCAN.INPUT
    A9999883 20 JUN 2006    AUDE0.NMPROD.REXX.ANALYZER.SCAN.INPUT
    A9999884 20 JUN 2006    AUDE0.NMPROD.REXX.ANALYZER.SCAN.INPUT
    A9999885 26 MAY 2006    AUDE0.NMPROD.REXX
  F1=Help      F2=Split     F3=Exit      F4=Return    F5=Find      F6=Refresh
  F7=Backward  F8=Forward   F9=Swap                  F11=Right
```

Generate Reports

You can generate summary and detailed reports from the Generate Report option, and then display the reports online or save them to a data set.

The summary report provides sufficient information to help you prioritize the migration of your NetView REXX procedures. The detailed report provides additional information to help you understand what facilities your REXX procedures are using.

Specify your reporting criteria through the REXX Analyzer Report Criteria Panel.

Note: For more information about the fields on the REXX Analyzer Report Criteria panel, see the online help.

Summary Report

A summary report provides the following information:

- Report input details
- Information to prioritize your Tivoli NetView REXX migration

Detailed Report

A detailed report provides the following information:

- Report input details
- REXX source error list
- GREXX compatibility issues
- REXX statement usage
- ADDRESS SUBCOM name
- Normalized command name (in ascending order by name)
- Normalized command name (in descending order by number of referencing procedures)
- Full command list
- External procedure references (in ascending order by name)
- External procedure references (in descending order by number of referencing procedures)
- External function references (in ascending order by name)
- External function references (in descending order by number of referencing procedures)
- INTERPRET statement expressions
- Pipe stage usage (in ascending order by name)
- Pipe stage usage (in descending order by number of referencing procedures)
- Information to prioritize your Tivoli NetView REXX migration

Example: Produce Online Reports

The following example shows how to produce an online report.

In this example, you process a single input REXX procedure library (AUDE0.QAREXXAN.EXEC). The library contains 31 REXX procedures. Some include deliberate syntax errors, unrecognized commands, functions, or pipe stages statements.

Note: The reports shown in this section may not be typical reports, but they illustrate most of the principal features of the online report.

Perform the Analysis

To produce the sample output

1. From the REXX Analyzer : Primary Menu, enter **A**.

   The Analysis Criteria panel appears.

2. Specify **AUDE0.QAREXXAN.EXEC** in the Input Data Set field. Use the default values for the other fields.

   Important! Your region must have read access to the input data set.

   Press F6 (Action).

3. Press F6 (Confirm).

   A message appears on the Analysis Confirmation panel indicating the REXX Analysis completed successfully.

Generate an Online Summary Report

To generate an online summary report

1. Press F5 (GenReport) from the Analysis Confirmation panel. You can change the default suppression character, if required.
2. Press F6 (Action).
3. Press F6 (Confirm).

   An online summary report appears.

Sample Report

The report that you generate using the REXX Analyzer provides sufficient information to let you prioritize the migration of your NetView REXX procedures.

When you produce an online report, press F8 (Forward) to scroll forward and view the various report sections.

The sample summary online report contains the following sections:

- Report Input Details (see page 78)
- Information to Prioritize Your Tivoli NetView REXX Migration (see page 79)

Report Input Details

The Report Input Details section shows the details of the resources used as input to the report:

```
                          Report Input Details
                          ____________________

ID:                       NETVIEW

REXX Procedure Data Set:  AUDE0.QAREXXAN.EXEC

Analysis Input Data Set:  AUDE0.NMPROD.REXXAN

Analysis Member:          A9999839

Date Analyzed:            26 MAR 2006

Time Analyzed:            20:20:34

Analyzed on System:       SYS1

Members Scanned:          31

Members Analyzed:         31
```

ID

Displays the report ID specified during report generation.

REXX Procedure Data Set

Identifies the REXX procedure data set that was analyzed.

Analysis Input Data Set

Identifies the data set in which the analysis output was saved.

Analysis Member

Identifies the member generated during the analysis that is used as input to this report.

Date Analyzed

Identifies the date the analysis process was run.

Time Analyzed

Identifies the time the analysis process was run.

Analyzed on System

Identifies the system on which the analysis was performed.

Members Scanned

Displays the number of procedures in the REXX procedure data set that were scanned during the analysis process.

Members Analyzed

Displays the number of procedures that passed analysis scan processing and were analyzed.

Information to Prioritize Your Tivoli NetView REXX Migration

This section of the report provides the following information that helps you to prioritize the migration of your NetView REXX procedures:

Results at a Glance

Displays the number of procedures that passed analysis and the number of procedures that failed.

Example: Results at a Glance

```
Results at a Glance
___________________

 22 procedures passed analysis.
 8 procedures failed analysis.
```

Explanatory Notes

Explains the information in the remainder of the summary report.

Recognized Command List

Identifies the NetView REXX commands that are emulated.

Recognized Pipe Stage Command List

Identifies the NetView REXX pipe stage commands that are emulated.

Recognized External Functions List

Identifies the NetView REXX functions that are emulated.

List of Procedure that Pass Analysis

Lists the recognized NetView REXX entities that have been emulated. Some procedures that contain recognized entities may still not run and could require updating to account for unsupported operands and GREXX compatibility issues.

Example: List of Procedures that Pass Analysis

```
List of Procedures that Pass Analysis (Total=22)
________________________________________

     DEBUG      SETU0001   TDRCI01    TDRCI04    TDRCI05
     TDRCS01    TDRSE01    TDRXF01    TDRXF10    TDRXF20
     TDRXF21    TDRXF22    TDRXP01    TDRXP10    TDRXP50
     TDRXP51    TDRXP52    TDRXP60    ZSLECHGF   ZSLEPAR
     ZSLTRACE   ZSWELDG
```

List of Procedures that Fail Analysis

Lists procedures that contain unrecognized entities. You can use this list to prioritize procedure updates.

Example: List of Procedures that Fail Analysis

```
Unrecog.  Procedure Entity
_____________________

    1     PNS0002      Commands: %LISTA

    1     SUB001       Commands: AUTOTASK

    1     TDRSE03   Pipe Stages: SQL

    1     TDRXP20   Pipe Stages: UNIX

    1     TDRXP70      Commands: @EXTRN01

    1     ZSLEAAGD     Commands: EZLEASLN

    3     CNMU0001     Commands: ASSIGN     AUTOTASK   START

   13     PNS0001      Commands: ACQ        ACTION     ADAPTER    ALERTSD
                                 AON        AONENABL   APPN       ASSIGN
                                 ATTACH     BGNSESS    BRIDGE     BROADCAST
                                 CANCEL
```

Unrecog.

Displays the number of unrecognized entities in a procedure.

Procedure

Identifies the procedure containing the unrecognized entities.

Entity

Identifies the entities that are not recognized. One or more of the following labels can be displayed:

- Commands—This label is displayed if a procedure contains commands that are not recognized.
- Functions—This label is displayed if a procedure contains external functions that are not recognized.
- Pipe Stages—This label is displayed if a procedure contains pipe stages that are not recognized.
- Usage—This label is displayed if usage information is available for a procedure. This number represents the number of hits weighted by time (as reported by the Tivoli NetView LIST MEMSTAT command).

Chapter 6: REXX External Assembler API

This section contains the following topics:

REXX and Assembler Programs

NetMaster REXX, like TSO/E REXX, provides an interface that contains the following features:

- You can write programs that provide new ADDRESS (Subcommand) environments.
- You can write external procedures and functions.
- These programs can access certain REXX facilities.
- You can write programs that can be called via the REXX ADDRESS LINK and ADDRESS ATTACH statements.

The interface is a subset of the interfaces described in the *TSO/E REXX Reference*:

- The interface is one-way. That is, REXX programs can *request* external interface services. However, external programs *cannot* request REXX execution. (For example, IRXEXEC and IRXJCL are *not supported*).
- The interface *can* make requests to REXX services that are normally available to external programs. For example, IRXSAY and IRXEXCOM are available.
- Some facilities are not available, or are not fully supported.

Environmental Considerations

In the TSO/E implementation of REXX, each user has their own address space. While (say, under ISPF), there may be more than one REXX process or task executing at the same time, the entire address space and all its resources are private to that user.

Under TSO/E, most programs execute *unauthorized*. The programs are loaded from non-authorized libraries, and cannot perform authorized functions. A program cannot do damage to system data areas.

Under Tivoli NetView, multiple users share the one address space. Work for different users is processed under different tasks. Also, the address space itself and all tasks in the address space execute in an APF-*authorized* environment.

Under CA NetMaster, multiple users share one address space, and the environment is APF-authorized. The ramifications of this, when writing REXX external interface programs, include the following:

- All programs must be loaded from an APF-authorized library. APF libraries are generally restricted for update by the security system.
- The interface programs can perform functions that may damage the address space or system. Thus it is imperative that they are carefully checked.
- User-written external interface programs must be registered before they can be used. This additional step prevents accidental execution of programs in an APF-authorized environment.
- It is a good idea to restrict update access to the REXX source libraries available to a region.

Make Programs Accessible to CA NetMaster

REXX external interface programs must be made accessible to CA NetMaster. This means that they must be in a STEP/JOB library or in the LINKLIST or LPA libraries.

CA NetMaster requires APF authorization. Because *all* libraries in the STEP/JOB library concatenation must be APF-authorized, this means that these external programs must be placed in an APF-authorized library.

Placing modules in an APF-authorized library is normally restricted by installation security configuration. Having placed the programs in an appropriate library, the programs must (in most cases) also be *registered* with NetMaster REXX.

Depending on the use of the program, this can be done as follows:

- If a program is to be called as an external PROCEDURE or FUNCTION by using the RXCTL DEFINE command:

  `RXCTL DEFINE FUNCTION=`*`funcname`* `[ LOADMOD=`*`modname`* `]`

  (If the LOADMOD operand is omitted, the function or procedure name is also the load module name).

- If a program is to be called as an external subcommand handler by using the RXCTL DEFINE command:

  `RXCTL DEFINE SUBCMD=`*`envname`* `LOADMOD=`*`modname`*

- A program that has been defined using one of the above methods may use the IRXSUBCM facility to add another program to the subcommand table. In this case no additional registration is required.

- Programs called using ADDRESS LINK and ADDRESS ATTACH do not need registration (But of course, they must be in an accessible library).

External Program Environment

Multiple concurrent user sessions are supported. A user session can concurrently execute any number of REXX processes. Any number of these REXX processes may request external programs be called.

To provide isolation between REXX processes, users, and so on, each REXX process that requests external programs, will have a private task attached. This task will be used to call *all* external programs, and so on, including CA product interfaces.

The task is *persistent*. Once started, it will stay running until the REXX process is terminated, or an ABEND occurs in an external program.

The external program can use REXX services (such as IRXUID) to obtain the user ID of the owning user, if this is important (for example, to issue RACROUTE requests).

Note: The task has *not* had an ACEE representing the user anchored.

By using a separate task, any waits, delays, loops, and so on, in the external program will not impact other users.

Note: If the REXX process terminates, the subtask will be detached. Any resources allocated by the external program will be cleaned up, for example GETMAIN'd storage (as long as it is not in a shared subpool). For example, if the external program has attached further tasks, these will be force-terminated (A03 ABENDs). Resources not associated with tasks (for example, dynamically allocated data sets) will stay allocated.

Provided Control Blocks

The following TSO/E REXX control blocks are emulated:

- Environment block (ENVBLOCK)
- Table of external entry points (EXTE)
- Workblock extension (WBEXT)
- Instorage block (INSTBLK)
- Parameter block (PARMBLOK), including:
  - SUBCOM table
  - Package Table (PACKTB)
  - Module Name Table (MODNAMET)
- Evaluation Block (EVALBLOK)

Environment Block (ENVBLOCK)

The environment block (ENVBLOCK) is supported as follows:

- The ID, version (0100), and length are all set correctly
- The parmblock pointer is set to point to the emulated PARMBLOCK.
- The USERFIELD is used internally. *Do not use this field*. (Because IRXINIT is not supported, this is of no consequence.)
- The WORKBLOK_EXT field is set to the address of the emulated WORKBLOK_EXT.
- The pointer to the table of external entry points is set to the emulated EXTE address.

Other fields are not set.

Table of External Entry Points (EXTE)

The table of external entry points is set to point to routines for *all* documented REXX external API facilities; however, some of these routines do not perform any useful function.

Workblock Extension (WORKBLOCK_EXT)

The workblock extension is supported as follows:

- The pointer to the instorage block (INSTBLK) is set.
- The WORKEXT_WORKAREA field is used internally. *Do not use this field*.

Other fields are not set up.

Instorage Block (INSTBLK)

NetMaster REXX programs are always loaded (and compiled) by CA NetMaster; however, the instorage block provides information about the program. For this reason, the following fields in the instorage block are provided:

- The acronym and header length.
- The pointer to the pointer table. A length of (8) indicates that one line address or length pair, or both, is present.
- The member name, ddname, and subcom names.
- The dsname field, which is set to a question mark (?).

Parameter Block (PARMBLOK)

The parameter block and related control blocks are supported as follows:

- The PARMBLOCK_ID field has a value of GRXPARMS (This distinguishes it from a standard IBM-format PARMBLOK).
- The version and language (ENU) are set.
- The module name table address is set.
- The subcom table address is set.
- The package table address is set.
- The flags are all zero, except that the flag and mask for the STORAGE function are set to indicate the function is disabled.
- The subpool value is set to 150 (An invalid subpool number).
- The parse token and address space name are (normally) set to NM.

Note: Unless otherwise noted, *do not alter these areas*.

The module name table is set as follows:

- All routine names are set to dummy names that cause a load failure if used. (Check that the EXTE table has valid addresses). You do not need to load any of these routines.
- The INDD and OUTDD fields have special values that are recognized by the IORTN for reading and writing to the user input and output devices (normally the user terminal).

The Subcommand table (SUBCOMTB) is set as follows:

- Each task has a separate copy of the subcom table (because the external programs must be able to alter their private areas).
- A subcom table header is built, as per the IBM documented format.
- Table entries, containing the subcommand environment name, handling module, and a blank token area, for all defined external subcommand environments, are built. (These entries are the CA external product interfaces, as well as any registered external user subcommand environments)
- Several 'available' entries are built (that can be updated by the SUBCOM facility).

A dummy package table (PACKTB) is built, with a header containing all zero counts.

Evaluation Block (EVALBLOK)

EVALBLOK is used for external procedure and function calls. One is provided, and the IRXRLT routine can obtain a larger one as required.

Supported Execution Interfaces

The interfaces described in the following sections are available for executing 'external' programs from NetMaster REXX.

Subcommand Handlers

These programs are executed when an ADDRESS statement or implied command statement is executed, and the specified (or current) ADDRESS subcommand environment is one that is an external program.

The following routines are called as documented in the *TSO/E REXX Reference*:

- R0 on entry points to the Environment Block.
- R1 on entry points to a parameter list.
- R13 (save area address), R14 (return address), and R15 (Entry point address) are standard.
- A mode will always be 31.

The parameter list is as follows (These are the descriptions of the parameters pointed to by the words in the parameter list pointed to by R1):

- Name of the host command environment (8 chars)
- Address of the command string (4 bytes)
- Length of the command string (4 bytes)
- Address of the user token in the SUBCOM table entry
- Word that can be set to the return code of the command (4 bytes)

External Procedure and Function Handlers

These programs are executed when a REXX CALL statement or function call (in an expression) refers to a procedure or function name that is not built-in, and is registered as an external user program.

The following routines are called as documented in the *TSO/E REXX Reference*:

- R0 on entry points to the Environment Block.
- R1 on entry points to a parameter list.
- R13 (save area address), R14 (return address), and R15 (Entry point address) are standard.
- A mode will always be 31.

The parameter list is as follows (These are the descriptions of the parameters pointed to by the words in the parameter list pointed to by R1):

- Reserved
- Reserved
- Reserved
- Reserved
- Input Parameter list. An array of 8-byte entries, each consisting of the address and length of a parameter. The list is terminated by 8 bytes of X'FF'.
- EVALBLOK. This block is used to return the value of a function call. (Note that if IRXRLT is called, this address may become invalid).

LINK and ATTACH Handlers

These programs are executed by the ADDRESS LINK pgm and ADDRESS ATTACH pgm statements.

The following routines are called as documented in the *TSO/E REXX Reference*:

- R0 on entry points to an Environment Block
- R1 on entry points to a parameter list
- R13 (save area address), R14 (return address), and R15 (Entry point address) are standard.
- Amode will be that of the target load module. But note that parameters will be above the line. Calling programs in AMODE 24 will probably result in an ABEND.

The parameter list is as follows (These are the descriptions of the parameters pointed to by the words in the parameter list pointed to by R1):

- Address of the passed character string
- Length of the passed character string

Entry Points in the EXTE Table

The table of external entry points (EXTE) is fully formatted. A routine address is present for *all* documented routines.

Note: This table *must not* be updated.

Not all routines are supported, and for those that are, not all of their functions are supported.

In general, an unsupported routine, or unsupported function, does *not* abend, but returns a severe error return code (generally 20). In addition, a message is sent to the CA NetMaster log.

The routines are supported as follows:

IRXINIT

All calls result in an error.

IRXLOAD

All calls result in an error.

IRXEXCOM

All standard documented functions are supported.

Note: GREXX only supports variable value lengths up to 32000.

IREXEXEC

All calls result in an error.

IRXINOUT

Some functions are supported:

- The INIT, TERM, and CLOSE requests are recognized, and ignored (RC = 0).
- The WRITE request is honored if the ddname is equal to the name in the MODNAMET table output ddname, and the output is treated as SAY output.
- WRITE to any other ddname results in an error return (RC=20), but no message is logged.
- Other requests result in an error and a message being logged.

IRXJCL

All calls result in an error.

IRXRLT

Only the GETBLOCK function is supported.

IRXSTK

Only the PUSH and QUEUE functions are supported

IRXSUBCM

All functions are supported, including ADD, DELETE, UPDATE, and QUERY, however, the subcom table that is referenced contains only entries for defined external subcommand environments, Also, there are a limited number of free entries that can be added to (any alterations made by IRXSUBCM are private to the associated REXX process).

IRXTERM

All calls result in an error.

IRXIC

All calls result in an error.

IRXMSGID

All functions are supported; however, there are no function codes. The routine always returns RC = 0, meaning display message ID.

IRXUID

All functions are supported, including USERID and TSOID, which return the associated user ID.

IRXTERMA

All calls result in an error.

IRXSAY

All functions are supported, including WRITE and WRITEERR, where the request is treated the same as SAY from the REXX program.

IRXERS

All calls result in an error.

IRXHST

All calls result in an error.

IRXHLT

All calls result in an error.

IRXTXT

All functions are supported, including DAY, MTHLONG, MTHSHORT, and SYNTXMSG.

The day and month functions return the standard values (in English).

The syntax messages returned are the GREXX versions.

IRXLIN

All functions are supported, including LINESIZE function, which returns the current logical terminal width. Background environments normally return 80.

IRXRTE

All calls result in an error.

Chapter 7: Executing NetMaster REXX Procedures Using NCL

This chapter provides examples for executing NetMaster REXX procedures using NCL.

This section contains the following topics:

Execution of REXX Procedures from an NCL Procedure

If you are using NCL procedures, you can use the REXX and NV commands to execute REXX procedures using NCL.

Example: Execute a REXX Procedure in Your Product Environment Using NCL

The following example shows the format of an NCL statement that executes a REXX procedure in your product environment:

```
REXX rexx_procedure parm_1 … parm_n
```

Example: Execute a REXX Procedure in the Tivoli NetView Emulation Environment Using NCL

The following example shows the format of an NCL statement that executes a REXX procedure in the Tivoli NetView emulation environment:

```
NV rexx_procedure parm_1 … parm_n
```

Execution of REXX Procedures Through Trouble Tickets

The alert monitor in your product region lets you use an NCL procedure to deliver trouble tickets in response to alerts, with or without operator intervention. The NCL procedure can include REXX commands that execute REXX procedures.

Appendix A: NCCF Commands

This appendix describes NCCF commands and parameters that are supported in CA NetMaster.

This section contains the following topics:

Get Help in the NCCF-like Facility

The NCCF-like facility in OCS provides online help on NCCF commands, REXX functions, and so on.

To get help in the NCCF-like facility

1. Enter **NCCF** at the command prompt in OCS.

   The following message appears:

   ```
   NCCF-like facility active, press PF3 to exit
   ```

2. Enter **HELP** followed by the command or function at the command prompt.

   The requested online help appears.

Note: You can also access the online help directly from OCS or Command Entry by entering NV HELP.

Supported NCCF Commands

The following NCCF commands and parameters are supported.

Note: The online help in the NCCF-like facility contains more information about the supported NCCF commands and parameters.

ACT

Activates the VTAM resource.

Parameters: *resname,* COMP, ALL, ONLY, SYNTAX, U, *passthru*

AFTER

Allows the operator to schedule a command or command procedure to run after a specified period of time.

Parameters: *time*, ROUTE, ID, TIMEFMSG, PPT, *commandtex*

ALLOCATE

Dynamically allocates a new or existing data set from the region.

Parameters: BLKSIZE, CATALOG, COPIES, CYLINDERS, DATACLAS, DATASET, DELETE, DEST, DIR, DSORG, FILE, HOLD, KEEP, LIKE, LRECL, MGMTCLAS, MOD, NEW, NOHOLD, OLD, RECFM, RELEASE, SHR, SPACE, STORCLAS, SYSOUT, TRACKS, UCS, UNIT, VOLUME, WRITER

AT

Schedules a command or command procedure to be run at a specific time.

Parameters: *time*, ROUTE, ID, TIMEFMSG, PPT, *commandtext*

BFRUSE

Performs the normal fully expanded Display VTAM Buffer VTAM command, displaying information about VTAM buffer use.

Parameters: Standard VTAM parameters.

CALC

Performs calculator functions using the REXX interpreter. It supports both decimal and hexadecimal calculations. The results are displayed in both decimal and hexadecimal (when available).

CCDEF

Reads and updates the Tivoli NetView Emulation CCDEF table.

Parameters: QUERY, MEMBER, DELETE

CLEAR

Clears the NCCF screen.

CLOSE

Ends all activity.

Parameters: NORMAL, STOP, IMMED, DUMP

CMD

Queues a command for processing.

Parameters: *commandtext*

DATE

Displays the current date and time.

DELAY

Waits for the specified period before sending the command or command procedure to the system.

Parameters: *time, commandtext*

DISPFK

Displays current values assigned to PFKs as defined for the NCCF emulation. The display information is in line mode rather than full screen mode.

DISPLAY

Performs the standard Display VTAM command.

Parameters: Standard VTAM parameters.

EVERY

Schedules a command or command procedure to be processed repeatedly at a timed interval.

Parameters: *time*, ROUTE, ID, TIMEFMSG, PPT, *commandtext*

EXCMD

Enables you to queue a command to another user. If the user is not logged on, then the command fails.

Parameters: *opid*, *commandtext*

FREE

Dynamically deallocates a data set from the region.

Parameters: FILE, DELETE, KEEP

GLOBALV

Enables you to define, save, and retrieve common or task global variables.

Parameters: DEFC, DEFT, PUTC, PUTT, GETC, GETT, *variable_name_list*

GO

Resumes running a command procedure that is in PAUSE status or WAIT status. You can use the GO command to give values to a command procedure that is in PAUSE status.

Parameters: ID

HELP

Provides online help for the NCCF-like emulation.

Parameters: null, REXX, REXX *itemname*, NCCF, NCCF *commandname*, PIPE, PIPE *pipename*, *itemname*, *message_id*

INACT

Inactivates the VTAM resource.

Parameters: *resname*, IMMED, FORCE, UNCOND, REACT, GIVEBACK, *passthru*

INACTF

Forces the VTAM resource to become inactive.

Parameters: *resname*, *passthru*

INPUT

Opens a full-screen window to facilitate the creation and editing of the multiline command entry buffer.

Parameters: *number_of_lines*

LIST

Displays information related to the specified parameter.

Parameters: TIMER=ALL OP=ALL, TIMER=*tid* OP=ALL, TIMER=ALL, STATUS=OPS, CLIST=*membername*, DSILOG

LISTA

Displays the data set status, disposition, ddnames, and data set names of the files currently allocated to the region. It can also indicate which data sets contain a specific member.

Parameters: *ddname*, *membername*

LOGOFF

Terminates the NCCF-like session.

MODIFY

Performs the standard Modify VTAM command.

Parameters: Standard VTAM parameters.

MSG

Sends specified text to specified target.

Parameters: ALL, LOG, *operid,* SYSOP

MVS

Enables you to enter a z/OS system operator command.

Parameters: *commandtext*

NETVASIS

Provides a way to enter a command in mixed case.

Parameters: *commandtext*

NM

Enables you to enter a CA NetMaster command.

Parameters: *commandtext*

PIPE

Provides most pipe stages supported in Tivoli NetView. These stages are intended to aid in the migration from NetView.

Options: STAGESEP, ESC, END, NAME, DEBUG 1

PURGE

Purges all timers for the user running the command or a specified timer.

Parameters: TIMER=ALL, TIMER=*tid*

RECYCLE

Inactivates and then reactivates the VTAM resource.

Parameters: *resname*

RMTCMD

Allows you to execute NCCF commands on a remote system.

Parameters: SEND LU=*linkname*, LU=*linkname*, *commandtext*

SUBMIT

Submits a batch job, either from a data set (SEQ or PDS), or if data set omitted, from the data set allocated to the DSIPARM DD.

Parameters: *datasetname*, *datasetname* (*membername*), *dsiparm membername*

TSOUSER

Displays the status of the TSO user ID.

Parameters: *id*

VARY

Performs the standard Vary VTAM command.

Parameters: Standard VTAM parameters.

Supported Command Prefix Label Parameters

Limited support is provided for the use of labels to route commands to INMC-linked regions.

The label has the following format:

linkname: [*wait_time*] *command_text*

Note: The online help in the NCCF-like facility (see page 102) contains more information about the command prefix label and its usage considerations.

Determine Link Names

When you use a command prefix label to route commands to an INMC-linked region, you need to know the name of the link to that region.

To determine the names of the Inter-Network Management Connection (INMC) links, enter the **SHOW LINK** OCS command in your region.

Information about the links defined in your region appears.

Get Help About the Command Prefix Label

The NCCF-like facility in OCS provides online help on the command prefix label.

To get help about the command prefix label

1. Enter **NCCF** at the command prompt in OCS.

   The following message appears:

   ```
   NCCF-like facility active, press PF3 to exit
   ```

2. Enter **HELP NCCF** at the command prompt.

   The NCCF-like Support for NetView Emulation online help panel appears.

3. Enter **S** beside Command Label under For More Help. If necessary, press F8 to scroll down the panel.

   The online help for the command prefix label appears.

REXX Commands

You can execute the following REXX commands from a REXX procedure:

Note: The online help in the NCCF-like facility (see page 97) contains more information about REXX commands.

DOM

Removes a WTO message from one or more consoles.

Parameters: SMSGID

Returned Values:

100 Invalid length.

104 Invalid value

PARSEL2R

Extracts data from the character-string value of a variable and assigns the extracted data to one or more variables using rules called a parsing template. The parsing template specifies a list of symbols, patterns or character selectors, or a combination of these separated by blanks.

Parameters: *source_variable, parsing_template*

Returned Values:

0 Processing completed successfully.

100 There are not enough parameters.

104 Input buffer is blank.

108 The command list dictionary lookup of source failed.

112 Hexadecimal data in the template is not valid.

116 The command list dictionary update failed.

120 The trailing slash (/) is missing.

VIEW

Displays the full-screen panel from user-written REXX procedures.

Parameters: *dummy_name, panel_name,* INPUT|NOINPUT, END|NOEND, SWAP|NOSWAP, MSG|NOMSG

Returned Values:

0 Processing completed successfully.

4 Error reading or processing panel definition.

8 Panel containing comment lines only or katakana.

16 Invalid parameters or environment.

81 Invalid panel definition.

WAIT

Suspends processing of a command until the period has elapsed.

Parameters: WAIT *nn*, WAIT *nn* SECONDS, WAIT *nn* MINUTES

Return Values:

0 Processing completed successfully.

8 There are too many parameters.

12 A syntax error occurred.

WTO

Sends a message to one or more consoles.

Parameters: DESC, LINETYPE, MCSFLAG, ROUTCDE, SYSCONID

Return Values:

4 WTO was sent with truncated text.

16 Internal processing failed.

112 SYSCONID length was not valid.

116 SYSCONID value is not valid.

124 The command list dictionary update failed.

136 LINETYPE is not valid.

140 KEY length is not valid.

164 ROUTCDE value is not valid.

172 MCSFLAG value is not valid.

176 DESC value is not valid.

204 No message passed. Completed successfully.

WTOR

Sends a message to one or more consoles and requests a reply.

Parameters: MCSFLAG, ROUTCDE, SYSCONID

Return Values:

4 WTO was sent with truncated text.

16 Internal processing failed.

100 No message passed. Completed successfully.

112 SYSCONID length was not valid.

116 SYSCONID value is not valid.

124 The command list dictionary update failed.

136 LINETYPE is not valid.

140 KEY length is not valid.

164 ROUTCDE value is not valid.

172 MCSFLAG value is not valid.

Appendix B: REXX Functions

This appendix describes the REXX functions.

This section contains the following topics:

REXX Functions

The returned values for all REXX functions are as follows:

Note: Unlisted functions always return a null value.

APPLID

Returns the application program identifier for the OCS window in which the command list is running.

ATTENDED

Always returns 1 (attended).

AUTOTASK

Always returns 0 (not an autotask).

CGLOBAL

Returns the value of the named common global variable, if it exists. Null is returned otherwise.

CMDNAME

Returns the name by which the program was called.

CURSYS

Returns the 1-8 character current system name.

DCO

Always returns N (not unattended).

DISC

Always returns 0 (not disconnected).

DISTAUTO

Always returns 0 (not a distributed autotask).

DOMAIN

Returns the name of the current emulated Tivoli NetView domain, as specified in the DSIDMNK member under the DSIPARM ddname.

FNDMBR

0 - The specified member name was found.

4 - The specified member name was not found.

O - *cccccccc rrrrrrrr*

O - DSS_PDS OPEN failed.

cccccccc - SYS.RETCODE

rrrrrrrr - SYS.FDBK

F *cccccccc rrrrrrrr*

F- DSS_PDS QUERY_MEM failed.

cccccccc - SYS.RETCODE

rrrrrrrr - SYS.FDBK

LU

Returns the logical unit name for this operator terminal.

NETID

Returns the value of the VTAM SSCP name.

NETVIEW

Returns NV34 if no arguments are passed to the function; otherwise, a string argument of T return NetView V1R3 Emulation by NetMaster.

OPID

Returns user ID for no argument, O argument, and S argument.

Returns null for R argument.

Returns? for T argument.

OPSYSTEM

Returns the specific operating system under which CA NetMaster is operating.

PANEL

Always returns 0 (panel commands are not allowed).

STCKGMT

Returns as an 8-byte hexadecimal value (the current Greenwich mean time in store-clock format).

SUPPCHAR

Returns the suppression character used by CA NetMaster and the NCCF-like facility, as specified in the DSIDMNK member under the DSIPARM ddname.

SYSPLEX

Returns the 1-8 character name of the z/OS SYSPLEX where the command list is executing. Available when running under MVS/ESA Version 4 Release 2.2 or above.

TASK

Always returns OST (Operator Station Task is the type of task under which the command list is running).

TGLOBAL

Returns the value of the named task global variable, if it exists. Otherwise null is returned.

TOWER

Always returns 0 or null. Null if the string argument ends with an asterisk. Otherwise 0 is returned, indicating no tower or subtower is enabled.

TYPE

Always returns ENT (enterprise option).

VTAM

Function returns the version and release of VTAM as a 4- character string in the form VT*v*r, where *v* is the version number and *r* is the release number.

Note: The value of VTAM() is null if VTAM is not active.

VTCOMPID

Returns the 14-character VTAM component identifier.

Note: The value of VTCOMPID is null if the VTAM program is not active.

WEEKDAYN

Returns a numeric value indicating the day of the week.

1 = Monday

2 = Tuesday

3 = Wednesday

4 = Thursday

5 = Friday

6 = Saturday

7 = Sunday

Appendix C: Pipe Stages

This appendix describes pipe stage commands and parameters, and edit stage orders that are supported in CA NetMaster.

This section contains the following topics:

Pipe Stage Commands

The following pipe stage commands and parameters are supported:

Note: The online help in the NCCF-like facility (see page 97) contains more information about the pipe stage command parameters.

BETWEEN

Divides message streams into sections.

Parameters: INCL, NOINCL, *number, position.length, /1st string/, /2nd string/*

CASEI

Compares character strings without respect to case.

Parameters: *stage_specification*

CHANGE

Replaces occurrences of one string with another.

Parameters: *position.length, /1st string/, /2nd string/, number*

CHOP

Truncates lines after a specified character, column, or string.

Parameters: *column, width, offset, /string/*, AFTER, ANY, BEFORE, NOT, STRING

COLLECT

Creates a multiline message, or messages, from input lines.

Parameters: AFTER, ASBEFORE, AT, BEFORE, BREAK, MAX, *number, position.length, /string/*

CONSOLE

Displays the contents of the pipeline.

Parameters: DUMP, XDUMP

CORRCMD

Processes a command with correlated wait and termination stages.

Parameters: ECHO, MOE, *wait, cmdlabel, commandtext*

CORRWAIT

Allows asynchronous messages into the pipeline.

Parameters: MOE, *, *interval*

COUNT

Counts the number of messages, lines, or bytes in the input stream.

Parameters: MESSAGES, BYTES, EACHLINE, EACHMSG, FROM, *number,* LINES, MAXLINE, MINLINE

DELDUPES

Deletes duplicate messages.

Parameters: ALL, KEEPFIRST, KEEPLAST, PAD, *position.length*

DROP

Specifies the number of messages to discard from the pipeline.

Parameters: *count,* FIRST, LAST, LINES, MSGS

DUPLICAT

Copies messages in the input stream.

Parameters: *number, **

EDIT

Creates or reformats messages.

Parameters: *hexstring, position.length, number, /string*

ENVDATA

Outputs environment data.

FANIN

Merges multiple input streams, in stream order, into a single output stream.

FANINANY

Merges multiple input streams, preserving the message order, into a single output stream.

FANOUT

Passes a single input stream to multiple output streams.

HOLE

Discards the contents of the pipeline.

JOINCONT

Joins consecutive messages that match a specified string.

Parameters: LEADING, NOT, */string/*, TRAILING

KEEP

Defines a task global place to store messages.

Parameters: *keepname,* APPEND, SEIZE

LITERAL

Inserts text into the pipeline.

Parameters: */string/*

LOCATE

Selects messages that match a specified character string to remain in the pipeline.

Parameters: ALL, FIRST, LAST, *position.length, /string/*

LOGTO

Sends a copy of the contents of the pipeline to a specific log.

Parameters: *, ALL, HCYLOG, NETLOG, SYSLOG

LOOKUP

Matches data within a pipeline

Parameters: APPEND, *detail_position.length, reference_position.length*, WILDCARD

MEMLIST

Creates a list of members in one or more partitioned data sets (PDS) or data definitions (DD).

Parameters: (DD), (DSN), *datasetname, ddname*

MVS

Runs specified MVS commands.

Parameters: *commandtext*

NETVIEW

Runs specified NetView REXX commands.

Parameters: *commandtext,* ECHO, MOE, NOPANEL

NLOCATE

Discards messages that match a specified character string.

Parameters: *position.length, /string/*

NM

Specifies to run a CA NetMaster command. The resulting messages are placed in the pipeline.

Parameters: *commandtext*

NOT

Changes the way output is treated by those stages that discard part of their output.

Parameters: *stage_specification*

PICK

Selects messages to remain in the pipeline based on a comparison of two strings.

Parameters: *position.length, operator, position2.length2*, or */string*

PIPEND

Causes a pipeline to end and issues a return code.

Parameters: *, *number*

PRESATTR

Changes the way messages appear on the Tivoli NetView console.

Parameters: *asis, color, highlighting, intensity*

QSAM

Reads from and writes to dynamically allocated data definition names or data sets.

Parameters: *data_set_name, data_definition_name,* (DD), (DSN)

REVERSE

Changes the order of message text and message lines.

Parameters: LINE, MESSAGE, STREAM

SAFE

Reads or writes messages to a command procedure message queue.

Parameters: *, *name,* APPEND, SEIZE

SEPARATE

Breaks MLWTOs into multiple single-line messages.

Parameters: DATA, SEQUENCE

SORT

Sorts input stream messages.

Parameters: A, D, PAD, *position.length*

SPLIT

Divides a line of text into multiple lines.

Parameters: *count,* AT, AFTER, ANY, BEFORE, STRING, */string/*

STEM

Reads or writes records to or from command procedure variables.

Parameters: (COMMON), (TASK), APPEND, COLLECT, *stemRoot,* FROM, *number*

STRIP

Removes characters from the beginning or end of a message.

Parameters: BOTH, LEADING, TRAILING, TO, NOT, BLANK, */character set/, limit*

SUBSYM

Substitutes z/OS and user-defined system symbolics in messages in the pipeline.

TAKE

Specifies the number of messages to be kept in the pipeline.

Parameters: FIRST, LAST, *count,* LINES

TOSTRING

Ends the data stream when a specific character string is located.

Parameters: ALL|FIRST|LAST, INCL, NOINCL, *position.length, /string/*

VAR

Reads or writes records to or from command procedure variables.

Parameters: (COMMON), (TASK), *name*

VARLOAD

Sets variables to a specified value.

Parameters: (COMMON), (TASK)

VTAM

Runs specific VTAM commands in a local or remote domain.

Parameters: *commandtext,* ECHO, MOE, NOPANEL

XLATE

Accepts a message from its input stream, translates specified characters, and writes the message to its output stream.

Parameters: *position.length*, UPPER, A2E, COMMON, E2A, LOWER

<

Reads data from DASD into the pipeline.

Parameters: DSIPARM. *ddname,* *, *member*

$STEM

Same as STEM, plus reads or writes message attribute variables associated with specified data variables.

Parameters: (COMMON), (TASK), APPEND, COLLECT, *stemRoot,* FROM, *number*

$VAR

Same as VAR plus reads or writes message attribute variables associated with specified data variables.

Parameters: (COMMON), (TASK), *name*

Edit Stage Orders

The following edit stage orders are supported:

COPY

Copies one or more unread lines in a multiline message from input to output.

Type: Global

COPYREST

Copies all unread lines in a multiline message from input to output.

Type: Global

FINDLINE n

Changes the current line to the absolute line number indicated by the argument.

Type: Global

FINDLINE string

Advances the current line to the line containing the specified target string.

Type: Global

FWDLINE n

Moves the current line forward by the number specified.

Type: Global

LASTLINE

Resets the input to the last line of a multiline message.

Type: Global

NEXTLINE

Combination of WRITELINE and READLINE. WRITELINE takes what is currently in the text buffer and writes it to the output message as a line.

Type: Global

PAD

Defines the padding character to be used by all other orders.

Type: Global

PARSE

Defines how the WORD input order will count words.

Type: Global

READLINE

Provides the next line of a multiline message to the input orders.

Type: Global

RESET

Cancels all existing SKIPTO and UPTO orders.

Type: Global

SKIPTO

Redefines the logical start of the input line.

Type: Global

TOPLINE

Resets the input to the first line of a multiline message.

Type: Global

UPTO

Redefines the logical end of the input line.

Type: Global

WRITELINE

Writes all text built by the output orders to the output message.

Type: Global

CURRGMT

Specifies an 8-byte store clock value generated at the time the order is executed.

Type: Input

hexstring

Specifies a hexadecimal string.

Type: Input

lineattr

Specifies that the input is one of the line attributes of the current line.

Type: Input

LINESENDER

Specifies the name of the sender.

Type: Input

msgattr

Specifies that the input is one of the message attributes of the current message.

Supported attributes are IFRAUSDR, MSGCOUNT, and MSGORIGIN.

Type: Input

position.length

Specifies the subset of the input line to be processed. The subset is defined by specifying a starting character and the total number of characters.

Type: Input

/string/

Specifies a delimited character string.

Type: Input

WORD

Specifies the subset of the input line to be processed. The subset is defined by specifying a starting word and the total number of words.

Type: Input

B2C

Converts string of Boolean values to a character string.

Type: Conversion

C2B

Converts input to a string of Boolean values.

Type: Conversion

C2D

Converts input to a string representing a decimal number.

Type: Conversion

C2F

Converts input to a string representing a signed floating point number.

Type: Conversion

C2V

Converts a varying length string to a character string.

Type: Conversion

C2X

Converts input to a string representing its hexadecimal notation.

Type: Conversion

D2C

Converts a signed integer number into a full word.

Type: Conversion

DT

Assumes the input text is a store clock (STCK) and converts the value to a readable 17-character string for the local time zone in the format MM/DD/YY HH:MM:SS.

Type: Conversion

FOUND

Translates a null string into No and any other string into Yes.

Type: Conversion

F2C

Converts a signed floating point number into a double-word.

Type: Conversion

LEFT

Truncates or pads the input to the length specified. Characters are counted from the beginning, or left, of the input.

Type: Conversion

ODDBYTES

Alternately, keeps and discards the input data.

Type: Conversion

OPDT

Assumes the input text is a store clock (STCK) and converts the value to a readable 17-character string representing the date and time in the format MM/DD/YY HH:MM:SS.

Type: Conversion

RIGHT

Truncates or pads the input to the length specified. Characters are counted from the end, or right, of the input.

Type: Conversion

STRIP

Removes all padding characters from the beginning and end of the input.

Type: Conversion

STRIPL

Removes all padding characters from the beginning of the input.

Type: Conversion

STRIPR

Removes all padding characters from the end of the input.

Type: Conversion

SUBSTR

Selects a subset of the input data.

Type: Conversion

UPCASE

Translates the standard 26-character Latin letters to uppercase.

Type: Conversion

V2C

Converts input to a varying length string.

Type: Conversion

X2C

Converts character data to internal hexadecimal format.

Type: Conversion

YESNO

Converts a 1-byte field to the character string Yes or No.

Type: Conversion

ZDT

Assumes the input text is a store clock (STCK) and converts the value to a readable - character string for Greenwich mean time in the format MM/DD/YY HH:MM:SS.

Type: Conversion

COLOR

Sets presentation attributes for the output line.

Type: Output

LINETYPE

Defines the line type attribute of the output line.

Type: Output

NEXT

Specifies that the input is to be placed into the output without an intervening blank.

Type: Output

NEXTWORD

Specifies that the input is to be placed into the output with an intervening blank.

Type: Output

position

Specifies that the data is to be placed in the output line beginning at the character, indicated by position.

Type: Output

Appendix D: Hypothetical Case Study—Migrating from Tivoli NetView

This appendix provides a case study illustrating how to migrate from Tivoli NetView.

This section contains the following topics:

About the Case Study

This case study follows the activities of a fictitious CA customer.

Black Lion Telecom is a new CA NetMaster customer. They have just chosen the product to replace the SNA networking functions of Tivoli NetView. They use the CA NetMaster NetView REXX Emulation facilities to ease their migration from Tivoli NetView.

This case study shows the steps Black Lion Telecom takes to successfully replicate their current Tivoli NetView NCCF SNA management capabilities under CA NetMaster.

The Hypothetical Environment

Black Lion Telecom has many large IBM z/OS mainframe systems running a mixture of in-house and vendor products to deliver their core telecommunications applications and support their corporate business needs.

Their internal data network is mainly SNA-based, supporting communication between their large central data centers and many geographically remote regional control centers. They also have a small but growing z/OS IP network.

For the past fifteen years, Black Lion Telecom has been using Tivoli NetView to manage their mainframe SNA network. Many of the Tivoli NetView SNA management features are used, but the core of the network operations work is done with NCCF.

New corporate business and IT alignments have forced an evaluation and justification of long-term mainframe software costs at Black Lion Telecom. As a result of this evaluation, management has decided to change to CA NetMaster NM for SNA and CA NetMaster NA.

Composition of the Migration Team

Black Lion Telecom's IT management assemble a team to migrate their NCCF network management system from NetView. Between them, the team members share the following skills and experience:

- NetView REXX coding—Programmers analyze the procedures and make any necessary changes to the REXX procedures to ensure that they work under CA NetMaster.
- Network administration—Network administrators help with the procedure analysis, identify which CA NetMaster features are of use, and implement new NCL procedures to complement the REXX procedures. Because CA NetMaster is new to them, they are learning it themselves.
- z/OS network operations—Network operators are the current end users of the NetView REXX procedures. They test the migrated and new procedures, and verify that they are happy with the final function provided.
- Process analysis and technical writing—Analysts and writers update the Black Lion Telecom Data Center's network operations instructions to incorporate any changes resulting from the migration. They also produce new training material.

Black Lion Telecom also has a CA consultant assigned to them to help with the migration.

The Migration Plan

The team produces a migration plan, to include the following broad tasks.

Note: Everyone migrating from NCCF faces the same typical tasks.

1. Identify and Prepare

   Identify what procedures really need to be migrated. Prepare them for migration. Make sure that the procedures work *before* migrating them. (A little time spent on this at the start can save a lot of unnecessary effort later.)

2. Familiarize

   Examine and experiment with the standard diagnosis and monitoring functions of CA NetMaster. The team may be able to use some of these to replace functions that are currently implemented by NetView REXX.

3. Analyze

   Analyze the NetView REXX procedures that need to be migrated. The team uses the NetMaster REXX analysis program to report on which REXX procedures need changing, and why.

   This program compares the NetView REXX commands, pipe stages, and functions in use with those provided by the CA NetMaster NetView REXX emulation environment. This provides an initial broad survey of the effort required for the next steps.

4. Migrate

   Update and test until the procedures work correctly. This is an iterative process. Where you start and where you go next is guided by the output of the REXX Analysis report, run during the analysis step.

   a. Update procedures—Make code changes to the NetView REXX procedures to enable them to run successfully under CA NetMaster. For any individual procedure, code changes might range from none to extensive.

   b. Test all procedures—Test the NetView REXX procedures under the Tivoli NetView Emulation environment. If they do not work, correct them.

5. Verify

   Bring everything together. Test and verify the migrated REXX procedures and any new NCL procedures against the production network.

Identify the NetView REXX Procedures that Need Migrating

The team must run every NetView REXX procedure considered for migration through an analysis program and investigate any potential incompatibilities. NetView REXX procedures not requiring any changes must also be verified in the CA NetMaster NetView Emulation environment.

With a large number of procedures, the effort required to do this can quickly increase. It is less work and far simpler to sort the NetView REXX procedures before you commence any migration steps.

Firstly, the team must identify the procedures that need to be migrated. They then use the REXX Analyzer to produce a report that guides them through the work of updating the procedures. If they did not do a good job of sorting and they analyzed a lot of unneeded procedures, the report will be longer and more complicated than it has to be.

Typical Tivoli NetView sites have a mixture of useful and non-useful procedures. Black Lion Telecom has been using Tivoli NetView NCCF for many years, and has accumulated a large number of NetView REXX procedures, both distributed and developed in-house. Their DSICLD ddname points to a concatenation of procedure data sets of varied histories.

New staff members have incorrectly placed testing procedures in production data sets. Procedure naming conventions have changed over time. Procedure data sets for older versions of Tivoli NetView are concatenated at the bottom.

The migration team decides to start with only essential procedures. It is careful to not touch the production environment. It takes the following actions:

- The systems programmers prepare a single PDS, BLTNET.NETVIEW.PRODREXX, for input to the REXX Analyzer. Initially they copy *all* members from *all* concatenated DSICLD data sets to it.
- They leave the production DSICLD data sets untouched—these are still being used by production Tivoli NetView and will not be changed in any way.

- The Tivoli NetView programmers examine all of the members of BLTNET.NETVIEW.PRODREXX, and identify and delete the members that they do not need to migrate. These include the following procedures:
  - Procedures that are not REXX, including many old Tivoli NetView CLISTs
  - Testing procedures
  - One-off procedures for transient problems that have been fixed
  - Unrecognized procedures with no known authors
  - Multiple nearly identical copies of the same procedures with different names
  - Procedures with the same names that exist in different DSICLD concatenation levels where only the highest level copy is examined

Altogether, the team identifies 200 NetView REXX procedures that need to be migrated.

Prepare the NetView REXX Procedures for Analysis

NetMaster REXX is based on GREXX, which is the CA common REXX engine. It is a compiled implementation, with minor differences to TSO REXX.

While the REXX Analyzer has a choice of scan types, the most straightforward is a scan type of REXXONLY—the others are for more advanced and specialized use. Using REXXONLY, any procedure that does not have a REXX header is not processed by the analyzer.

Because the REXXONLY report from the analyzer is what maps out much of the migration work, the programmers edit all of the members of BLTNET.NETVIEW.PRODREXX. In each one, they add the following comment as the first line:

```
/* REXX */
```

More information:

REXX Source Recognition (see page 14)

Analyze the NetView REXX Procedures

A completed migration can look almost seamless to end users of NCCF; however, CA NetMaster NetView Emulation is technically different from native NetView REXX. Although the emulation includes many of the most popular Tivoli NetView commands, functions, and pipe stages, not everything will work unchanged. Obviously, less work is required to migrate NetView REXX code that already uses the more common coding options.

A careful analysis helps you estimate how well the NetView REXX procedures fit the NetMaster REXX environment. Analyzing the NetView REXX procedures is the heart of the migration effort. After all the theories and promises, the analysis answers the following question:

"How compatible is my specific NetView REXX code under the CA NetMaster NetView REXX emulation environment?"

This in turn helps you answer the next question:

"How much work will I have to do to change my NetView REXX code?"

Run the REXX Analyzer

The Black Lion Telecom migration team performs the REXX procedure analysis by using the REXX Analyzer.

The team has already implemented a CA NetMaster NM for TCP/IP region, NETM10, on LPAR SYSA. SYSA is in the same sysplex as LPAR SYSB, where the soon-to-be-replaced NETVIEW9 task runs.

The team takes the PDS of NetView REXX procedures prepared previously and inputs this to the REXX Analyzer using the following procedure.

To run the REXX Analyzer

1. Enter **/REXXAN.A** in a CA NetMaster region.

   The Analysis Criteria panel appears.

2. Update the fields as required, and then press F6 (Action).

```
Scan Type .............+ REXXONLY
Do you use CA-PDSMAN? .. YES    (Yes/No)

Member Details (Comma delimited)
 Include Member List .. ____________________________________________
                        ____________________________________________
 Exclude Member List .. ____________________________________________
                        ____________________________________________

Data Set Details
 Input Data Set ....... BLTNET.NETVIEW.PRODREXX
 Output Data Set ...... BLTNET.NETM10.REXXAN

NetView Details (REXX usage)
 System Name .......... SYSB
 Running STC Name ..... NETVIEW9
```

   The Analysis Confirmation panel appears.

3. Press F6 (Confirm).

   The analyzer analyzes Black Lion Telecom's 200 NetView REXX procedures and writes the output to a member of BLTNET.NETM10.REXXAN.

   The REXXONLY scan type ensures that only procedures with REXX in a comment block on the first line are included in analysis processing.

   The NetView Details (REXX usage) fields are optional. Because they are filled in, a NetView REXX SHOW MEMSTAT command is sent to the specified NetView REXX started task, and the results included in the analysis report. The NetView REXX started task should be active for a while for the usage figure to be meaningful.

Generate and Review the Report

The team uses the following procedure to generate a report from the output of the analysis.

To generate a REXX Analyzer Summary Report

1. Enter **/REXXAN.G**.

   The Report Criteria panel appears.

2. Specify SUMMARY in the Report Type field, update the other fields as required, and then press F6 (Action).

   The Report Confirmation panel appears.

3. Press F6 (Confirm).

   The report is displayed online or saved in a data set as specified.

The migration team look at the report. There is so much information. They are not sure where to start.

While there is no strict step-by-step formula for a migration—what you want and how you have implemented NCCF varies widely—Black Lion Telecom's CA consultant recommends the following approach that has worked for other successful Tivoli NetView migrations:

1. Move the easiest, most potentially compatible NetView REXX procedures over first. Do the changes needed to get them going quickly.
2. Examine the standard SNA Diagnosis and Monitoring features of CA NetMaster. These include NEWS (for SNA event management), NTS (for SNA session awareness), and NCPView (a status monitor for NCPs).

   Examine the remaining NetView REXX procedures to identify which of these provide functions that can be replaced by an existing feature.
3. Update and test the remaining, less compatible NetView REXX procedures. For each, determine and perform the changes required to run it in CA NetMaster. These procedures will fall into the following groups:
   - Those needing reasonably small REXX code changes
   - Those needing bigger REXX code changes
   - Those needing redevelopment in NCL

Because CA Technical Support has been involved in many successful migrations, it has considerable experience in the area of code changes, and can easily support the Black Lion Telecom migration team with this part of their migration.

Migrate REXX Procedures that Passed Analysis

The List of Procedures that Pass Analysis (Total=*n*) section near the end of the REXX Analyzer report identifies the REXX procedures that are most likely to be compatible. These procedures are usually the simplest to migrate. The team migrates these procedures first.

These procedures contain only recognized entities, which are the REXX commands, functions, and pipe stages that have been implemented by the CA NetMaster NetView REXX Emulation environment. This does not guarantee that they will work unchanged, but they will generally be simple to change if required.

Of Black Lion Telecom's 200 NetView REXX procedures, the report indicates that 100 of them are procedures that passed analysis.

Verify the Procedures in Tivoli NetView

The CA consultant recommends this step—in previous migrations, a lot of time was wasted attempting to fix migrated procedures that did not work originally under Tivoli NetView. If problems occur with a procedure under CA NetMaster, the fact that it was proven to work under Tivoli NetView narrows the problem determination effort.

The network operators note the names of the 100 procedures that passed analysis.

They go to their production NCCF consoles and run the 100 procedures in Tivoli NetView to verify that they work correctly *before*, as well as *after*, the move.

They find that two of the procedures do not work in Tivoli NetView. They decide not to migrate these procedures.

Test the Procedures in CA NetMaster

The migration team adds their NetView REXX procedures data set to NETM10's COMMANDS concatenation:

```
*-------------------------------------------------------------------
*      PRODUCT REGION COMMANDS DATASET ALLOCATION
*-------------------------------------------------------------------
        DD=COMMANDS,BLKSIZE=32000,DISP=SHR,DSN=BLTNET.NETM10.TESTEXEC
        DD=*,DISP=SHR,DSN=BLTNET.WHATEVER.CC11EXEC
        DD=*,DISP=SHR,DSN=BLTNET.WHATEVER.CC2DEXEC
        DD=*,DISP=SHR,DSN=BLTNET.NETVIEW.PRODREXX
```

Because the REXX procedures use a customized CCDEF table and the VIEW command, the team also updates the NETVEMLDSN parameter group in NETM10 to point to the Tivoli NetView system definitions and panels that are used by the procedures.

The team then run these 100 REXX procedures, one by one, unchanged, in NETM10.

They run the procedures using the NCCF Emulation Facility. To access this facility, they type the command, **NCCF**, from OCS.

This facility provides basic NCCF look-alike support. All commands entered under this facility are automatically assumed to be NetView REXX commands or procedures.

Example: Successful Procedure

Here is the result of running a procedure that checks the status of the currently active VTAM exits and reports if the required exit is not active.

```
(18.05)---------- NetMaster Operator Console Services (PROD1) -----------------
PROD1:USER01                        NCCF-like facility active, press PF3 to exit
ACTVTMX10  - CHECKING ACTIVE VTAM EXITS  - 28 Mar 2006 18:05:17

ACTVTMX10 - VTAM EXIT: ISTEXCUV IS ACTIVE
ACTVTMX10 - VTAM EXIT: ISTEXCSD IS ACTIVE
ACTVTMX10 - VTAM EXIT: ISTEXCAA IS ACTIVE
ACTVTMX10 - VTAM EXIT: ISTEXCGR IS ACTIVE
ACTVTMX10 - VTAM EXIT: ISTEXCPM IS ACTIVE
ACTVTMX80 ------------------------------------------------------------------
ACTVTMX80 - ** VTAM EXIT: ISTEXCVR IS REQUIRED BUT IS CURRENTLY INACTIVE **
ACTVTMX80 ------------------------------------------------------------------

ACTVTMX99 DONE
```

The procedure is as follows:

```
/* REXX */
MSGID   = 'ACTVTMX'
WARNING = 'NO'

  SAY MSGID||'00' ' - CHECKING ACTIVE VTAM EXITS  -' DATE('N') TIME('N')
  SAY
  'PIPE VTAM D NET,EXIT ',
  '  | CORRWAIT 10',
  '  | SEP ',
  '  | LOC /IST1251I/',
  '  | STEM ACTX.'

   DO I = 1 TO ACTX.0
     VTAMEXIT   = WORD(ACTX.I,2)
     EXITSTATUS = WORD(ACTX.I,5)
     IF (EXITSTATUS = 'ACTIVE') THEN
       SAY MSGID||'10' '- VTAM EXIT: 'VTAMEXIT 'IS 'EXITSTATUS
     ELSE
       IF (VTAMEXIT = 'ISTEXCVR') THEN
         WARNING = YES
   END

  IF (WARNING = YES) THEN
    DO
      SAY MSGID||'80' '-----------------------------------------'||,
                      '--------------------'
      SAY MSGID||'80' '- ** VTAM EXIT: ISTEXCVR IS REQUIRED BUT IS '||,
                      'CURRENTLY INACTIVE **'
      SAY MSGID||'80' '-----------------------------------------'||,
                      '--------------------'
    END

  SAY
  SAY MSGID||'99' 'DONE '
EXIT
```

Review Test Results

Of Black Lion Telecom's 100 NetView REXX procedures that passed analysis:

- 58 procedures execute without error the first time and produce the correct output.
- 12 procedures fail with a GREXX compile error.
- 10 procedures execute without error but produce incorrect output.
- 15 procedures have unrecognized or unsupported command parameters.
- 5 procedures failed during execution.

The team examines each of the procedures that did not work.

Note: When testing the REXX procedures that passed analysis, there is no such thing as a typical spread of results. There is a very wide variance, from site to site, in the percentage of those procedures that will have a particular outcome.

Correct and Retest the Procedures in CA NetMaster

The team corrects the procedures based on the encountered problem and retests them.

At the end of all of the repeated testing, updating and retesting of the procedures, 100 of Black Lion Telecom's 200 NetView REXX procedures are now running successfully in NETM10.

Procedures with GREXX Compilation Errors

Causes of compilation errors are usually simple to identify.

After examination, a few procedures that got GREXX compile errors are found not to be REXX procedures, but Tivoli NetView CLISTs. The REXX header had been incorrectly added to them, so they were analyzed by mistake. The procedures are either removed from analysis or rewritten in REXX.

The other GREXX compile errors have the same cause. The REXX procedures contain a SIGNAL ON ERROR statement but no corresponding label. NetView REXX tolerates this, but CA NetMaster does not.

Example: Procedure That Runs in Tivoli NetView but Does Not Compile in CA NetMaster

```
/* REXX */
  SIGNAL ON ERROR
  TOTAL = (1 + 1)
  SAY TOTAL
```

The following compilation error results:

```
(18.20)---------- NetMaster Operator Console Services (PROD1) -----------------
PROD1:USER01                      NCCF-like facility active, press PF3 to exit
--------------------------------------------------------------------------------
NR1101 REXX COMPILE OF TT2 IN LIBRARY COMMANDS HAS ERRORS/WARNINGS:
NR1284 ERROR 16 ...
NR9916 Error within TT@     , line 2 Label not found
```

The solution is to make the REXX procedure code more accurate. Add the labels and associated proper error processing logic, as dictated by good programming practice.

When a procedure with a missing ON ERROR label is run in Tivoli NetView and a run-time error occurs, the processing continues exactly as if the ON ERROR statement was not there. This is not an advantage—there could be a false sense of security from having SIGNAL statements that do nothing when triggered.

Procedures with Incorrect Output

Some NetView REXX procedures execute under CA NetMaster, but they produce incorrect output.

After examining the code, the REXX programmers found the problem is due to some message prefix differences between Tivoli NetView and CA NetMaster.

These procedures are updated to parse for the CA NetMaster messages prefixes.

Procedures with Unsupported Parameters

Some NetView REXX procedures fail to execute because their parameters are not recognized. Although these procedures passed analysis, not every parameter on every emulated NetView REXX function, command, or pipe stage is supported.

Some of the unsupported parameters referred to Tivoli NetView-specific things that are not applicable under CA NetMaster. These parameters are removed. Retesting shows no loss of function.

Some of the unsupported parameters do things that Black Lion Telecom requires. The REXX programmers update these procedures and use the appropriate NetMaster REXX pipe stage to issue CA NetMaster commands that perform the required function.

NetMaster REXX supports many common Tivoli NetView pipe stages. The REXX programmers use the online help to get details of the pipe stage command parameters.

From OCS, they type **NCCF**, press F1, and select PIPE (NCCF) command.

Example: Unsupported DELETE Parameter of CON Pipe Stage Command

Here is an example of running the following procedure that contains unsupported operands:

```
/* REXX */
  " PIPE LIT /HELLO/|CON DELETE "
EXIT
```

The following error results:

```
(18.48)---------- NetMaster Operator Console Services (PROD1) -----------------
PROD1:USER01                        NCCF-like facility active, press PF3 to exit
--------------------------------------------------------------------------------
XNEM005 Unsupported keyword argument: DELETE
XNH362E PIPELINE TERMINATED. ERROR IN STAGE 2 IN PIPELINE 'PIPE': CON DELETE
```

Procedures That Did Not Execute

Some NetView REXX procedures do not execute correctly.

After examining the code, the REXX programmers find that the procedures need non-executable files—data, tables, and lists of VTAM commands—that they usually find in the DSICLD data sets.

When the members were previously copied to BLTNET.NETVIEW.PRODREXX, the REXX procedures only were copied. This was so that only the very essential members were copied.

The non-executable members required by these procedures are now copied to the CA NetMaster PDS from the DSICLD data sets. The procedures now find their data and execute correctly.

Considerations

Unlike NetView REXX, where procedure compilation is additional and chargeable, NetMaster REXX is always compiled. In production, this is a performance benefit.

In a development situation, where a REXX procedure may be frequently modified and retested, programmers can disable automatic compilation by *one* of the following methods:

- Use the UNLOAD command to manually unload changed procedures.
- Use the PROFILE NCLTEST options to cause automatic recompilation.

Migrate REXX Procedures That Contain Only One Unrecognized Entity

The Unrecog. Procedure Entity section of the REXX Analyzer report identifies the procedures that contain only one unrecognized entity. For each procedure, the section identifies the NetView REXX command, function, or pipe stage that is not recognized. It also displays the usage of the procedures if the NetView Details (REXX usage) fields were filled in on the Analysis Criteria panel. It indicates the number of hits weighted by time for a procedure, from the Tivoli NetView SHOW MEMSTAT command.

All this information helps to plan the recoding effort to bring the greatest immediate rewards. If many procedures have the same single unrecognized entity, then learning how to replace the use of that one entity results in many working procedures. Similarly, if there are procedures with high usage figures, they also need to be concentrated on early.

Some unrecognized entities can be replaced by equivalent CA NetMaster facilities and do not require extensive code updates. In other cases, they do require extensive code updates.

Sometimes, updates involve updating the REXX code. In a few cases, the procedure will need the facilities of NCL.

The REXX Analysis report shows that of Black Lion Telecom's 200 NetView REXX procedures, 70 contain only a single unrecognized entity.

The report shows that the most frequent unrecognized entity is the Tivoli NetView TRAP command, which is typical for most migrations.

Example: Replace TRAP Command

In Tivoli NetView, you can use the PIPE command, or the TRAP and MSGREAD commands to process command output in REXX.

Procedures that use TRAP, WAIT, and MSGREAD tend to be older procedures that were developed before the availability of the corresponding pipe stages. IBM recommends against further use of superseded commands like these.

The following NetView REXX procedure uses the TRAP and MSGREAD commands:

```
/* REXX                                                              */
/*      Simple example on how to replace a TRAP/MSGREAD code segment  */
/*      in a NetMaster NetView emulation REXX procedure               */
/*                                                                    */
/*--------------------------------------------------------------------*/
/*  Part 1 - The NetView REXX procedure                               */
/*                                                                    */
/*           a) Issue the TRAP command                                */
/*           b) Issue the DATE command - The expected message is:     */
/*                CNM359I DATE : TIME = hh:mm DATE = mm/dd/yy         */
/*                                                                    */
/*           c) Read message (MSGREAD)                                */
/*           d) Extract and display the current time                  */
/*--------------------------------------------------------------------*/

  "TRAP AND SUPPRESS ONLY MESSAGES CNM359I "
  "DATE"
  "MSGREAD"
   SAY " THE CURRENT TIME IS: "||MSGVAR(5)

   SAY " BYE "
```

This Tivoli NetView procedure produces the following output:

```
NCCF                     Tivoli NetView   CNM01 USER01 01/04/06 10:01:06
 * CNM01    MYPROC
 C CNM01     THE CURRENT TIME IS: 10:01
 C CNM01     BYE
```

The following CA NetMaster procedure that replaces the Tivoli NetView procedure does not use the TRAP command:

```
/* REXX                                                              */
/*      Simple example on how to replace a TRAP/MSGREAD code segment  */
/*      in a NetMaster NetView emulation REXX procedure               */
/*--------------------------------------------------------------------*/
/*                                                                    */
/*  Part 2 - The NetMaster NetView Emulation REXX procedure:          */
/*                                                                    */
/*           The TRAP, DATE command, MSGREAD and result display have  */
/*           been replaced by PIPE command:                           */
/*                                                                    */
/*            a) Issue DATE command via NetView stage                 */
/*            b) Select XND359I message (NetMaster CNM359I equivalent)*/
/*            c) Extract the time                                     */
/*            d) Display the result                                   */
/*                                                                    */
/*--------------------------------------------------------------------*/
  "PIPE NETVIEW  DATE",
  "             |LOC /XND359I/",
  "             |EDIT /THE CURRENT TIME IS:/ NEXT WORD 6",
  "             |CON"
  SAY "BYE "
```

This procedure produces the following output:

```
(20.01)---------- NetMaster Operator Console Services (PROD1) -----------------
PROD1:USER01                      NCCF-like facility active, press PF3 to exit
--------------------------------------------------------------------------------
THE CURRENT TIME IS: 20:01
BYE
```

Migrate REXX Procedures That Contain Multiple Unrecognized Entities

Migrating procedures that have multiple unrecognized entities is similar to migrating procedures with a single unrecognized entity.

Of Black Lion Telecom's 200 NetView REXX procedures, 30 of them contain multiple unrecognized entities. They need to provide equivalent function in CA NetMaster.

These include some of the oldest REXX procedures and those that will need the most redevelopment and coding work.

Some of the procedures contain only the unrecognized entities that were encountered while migrating procedures containing a single unrecognized entity. Some of these can be updated in the same manner.

This leaves the more challenging procedures to migrate. Some of these procedures can be replaced by CA NetMaster features, thus minimizing recoding. Also, full-screen handling is already done by these features. Any remaining procedures can be converted with help from Technical Support.

CA NetMaster Features

CA NetMaster has a wide range of SNA management features.

The CA consultant has been working with the migration team to examine and identify where standard CA NetMaster NM for SNA and CA NetMaster NA options can replace functions implemented with NetView REXX.

The consultant advises that an easy way to access and review the CA NetMaster NM for SNA features is to enter **/SNA** at any menu prompt to list the high-level shortcuts to SNA features. The shortcuts can then be used to access those features.

Black Lion Telecom has several NetView REXX procedures that were written to do APPN-related tasks. These tasks, and more, are already done by the APPN features in CA NetMaster.

Example: APPN Features

The APPN Menu enables you to display the APPN Transport Resource List, which is typical of a CA NetMaster NM for SNA display. Additional data can be displayed by pressing PF11.

```
PROD1------------------ NCS : Transport Resource List -------------------------
Command ===>                                                  Scroll ===> CSR

     S/=View TRLE D=Display TRLE SU=View ULP PU SUC=View ULP CP ?=More Actions
   TRL                  MPC    MPC            --Upper Layer Protocol (ULP)--
   Entry    Ctrl Status Level  Usage    HPDT  PU       CP Name           TG
   AMF1G01Z MPC  ACTIV  QDIO   SHARE    YES
   AMF1G02X MPC  ACTIV  QDIO   SHARE    YES
   AMF1G03X MPC  NEVAC  QDIO   SHARE    *NA*
   AMF1G51X MPC  NEVAC  QDIO   SHARE    *NA*
   AMF1G52X MPC  NEVAC  QDIO   SHARE    *NA*
   AMF1G53X MPC  NEVAC  QDIO   SHARE    *NA*
   AMF3G10X MPC  NEVAC  QDIO   SHARE    *NA*
   A11TA01  MPC  ACTIV  NOHPDT ***N/A** NO    A11AP01  SERVRA01.A01X99   21
   A11TA29  MPC  ACTIV  NOHPDT ***N/A** NO    A11AP29  SERVRA01.A29X99   21
   A11TA31  MPC  ACTIV  NOHPDT ***N/A** NO    A11AP31  SERVRA01.A31X99   21
   A11TA55  MPC  ACTIV  NOHPDT ***N/A** NO    A11AP55  SERVRA01.A55X99   21
   A11TA57  MPC  INACT  NOHPDT ***N/A** *NA*
   A11TA58  MPC  ACTIV  NOHPDT ***N/A** NO    A11AP58  SERVRA01.A58X99   21
   A11TA61  MPC  ACTIV  NOHPDT ***N/A** NO    A11AP61  SERVRA01.A61X99   21
   ISTT11D1 XCF  ACTIV  HPDT   SHARE    *NA*  ISTP11D1 SERVRA01.A01X99   22
   ISTT1131 XCF  ACTIV  HPDT   SHARE    *NA*  ISTP1131 SERVRA01.A31X99   22
 F1=Help      F2=Split     F3=Exit                 F5=Find      F6=Refresh
 F7=Backward  F8=Forward   F9=Swap                F11=Right
```

A wide variety of actions can be performed against each entry. For example, entering S beside an entry opens the Node Display - TRLE panel:

```
Node Name ... AMF1G01Z                              Link Name ... PROD1

                          ____________
                         |   HOST     | NetID    SERVRA01
   (NONE)                |            | OpSys    z/OS
   **END**               |  A11X99    | Subarea  11
                         |____________| VTAM ... 6.1.7
                              ||
                          ____||_____
                         |   TRLE     | Status   ACTIV
                         |            |  Desired  ACTIV
                         | AMF1G01Z   | Linetyp  LEASED
                         |____________| Prcol    MPC
```

Another example is to use APING to test connectivity:

```
PROD1-------------------- NCS : APING Results List ---------------------------
Command ===>                                                 Scroll ===> CSR

 Resource Name .......  SERVRA01.A01X99
 Count ...............  3
 Packet Count ........  1
 Packet Size .........  100
 Echo? ...............  ___
 Logmode .............  ________
 Transaction Program  ____________________________________________________
    ------------------------------------------------------------------------
    Resource          Min/Avg/Max    Cnt Pkts  Size     Rate LogMode  COS
    USILDA01.A01X99   1/1/2            3    1   100  150KB/s #INTER   *BLANK*
    ------------------------------------------------------------------------
    IST1460I  TGN  CPNAME             TG TYPE      HPR
    IST1461I   21  SERVRA01.A01X99    APPN         ANR
    IST1463I  ALLOCATION DURATION:        5 MILLISECONDS
    IST1464I  PROGRAM STARTUP AND VERSION EXCHANGE:       2 MILLISECONDS
   **END**



 F1=Help      F2=Split     F3=Exit                                F6=Action
 F7=Backward  F8=Forward   F9=Swap     F10=Topology F11=VTAMDisp
```

Complete the Migration

Black Lion Telecom's NetView REXX procedures—some unchanged, some with minor updates, some rewritten, and a few totally replaced by NCL—are finally all running in NETM10.

NETM10 runs in parallel with NETVIEW9. The operation of NETM10 is repeatedly tested, compared, and verified against the Tivoli NetView system it will replace. It meets the NETVIEW9's existing service level agreements with no difficulty.

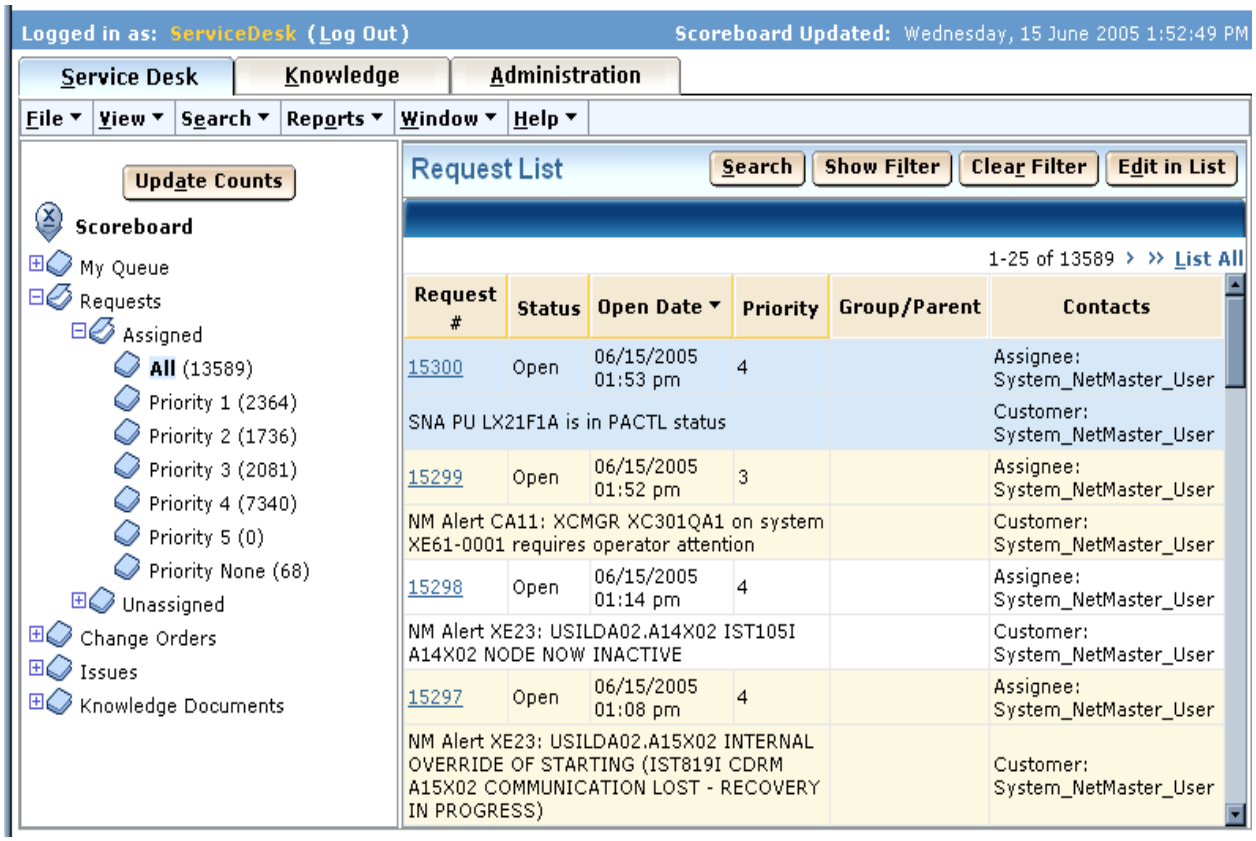
As part of the migration, the following tasks were performed:

- The network operations staff receives basic CA NetMaster training. Very little training is required in many areas because the migrated REXX procedures perform almost identically to their Tivoli NetView equivalents. The NCCF look-alike facility helps a lot.
- The technical writers update Black Lion Telecom's network operations procedures.
- The support staff learns to use the CA NetMaster Alert Monitor. Black Lion Telecom plans to make the Alert Monitor part of their standard monitoring.

```
PROD1  (20.02.08)------ Alert Monitor (22: 3 5 6 8 ) ----------Link: *MULTIPLE*
Command ===>                                                   Scroll ===> CSR

          S/B=Browse T=Track N=Notes A=Analyze TT=TroubleTicket C=Close ?=More
   Time     Description                              Resource          Track
   20.02.06 TCPIP11 over CPU baseline                TCPIP11
   19.57.43 NMTEST PING > 1 SEC                      172.24.122.87
   19.12.40 &$dfvalue                                RSRC04
   19.58.17 STACK RUNTCP: EXCP High, 120 > limit of  RUNTCP
   19.58.17 13244 1 TCPIP11                          TCPIP11
   19.52.43 PINGRTT High, 355ms > limit of 310ms     JM#232000
   19.30.21 SSI database freespace low: DB=TRACE, S  SSI
   11.15.37 Full dataset DSN00.PROD13.NTSLOG         NTSLOG
   19.58.17 STACK RUNTCP: TotalCPU High, 368ms > li  RUNTCP
   19.58.10 ASMON TCPIP11-TCP(2604): PortStatus is   PROD4
   19.53.08 ASMON TCPIP11-TCP(9999): PortStatus is   RSRC03
   07.32.06 STC JB on system CA11-0001 requires ope  RSRC02
   07.32.04 STC RSRC05 on system CA11-0001 requires  RSRC05
   07.31.50 FTSMON PROD13FT.PROD1 on system CO11-00  PROD13FT.PROD1
   19.12.40 57 data extents on AUDE0.PROD13.EVNTDB   EVNTDB
   19.12.37 114 data extents on AUDE0.PROD13.VFS     VFS
   06.12.22 7 index extents on DSN00.PROD13.IPLOG    IPLOG
 F1=Help      F2=Split     F3=Exit      F4=History   F5=Find      F6=Hold
 F7=Backward  F8=Forward   F9=Swap                   F11=Right
```

CA Service Desk Integration

Black Lion Telecom runs CA Service Desk. Because CA Service Desk requests can be created from alerts for serious network error conditions, help desk personnel are given contact procedures for z/OS SNA Network Support. The following Request List shows some requests.



What Is Next?

Black Lion Telecom's migration team have learnt a lot about CA NetMaster NetView REXX Emulation. Their NetView REXX programmers have an overview of the CA NetMaster development environment. Now that the migrated NCCF REXX procedures are stable, their network operations analysts have time to familiarize themselves with all the SNA management capabilities in CA NetMaster.

They begin to plan their further migration from NLDM to CA NetMaster NTS and from NPDA to CA NetMaster NEWS. They also plan to use NCPView and SNA Summary in CA NetMaster. This completes their Tivoli NetView migration.

Unlike NCCF, where you often did most of your customized development, migration to NTS, NEWS, NCPView, and other CA NetMaster SNA management features requires comparatively little if any coding. Robust, out-of-the-box menu structures are provided with these SNA features that let many ex-Tivoli NetView customers work in CA NetMaster quickly.

As Black Lion Telecom's z/OS IP network grows in the future, CA NetMaster NM for TCP/IP can help them.

Index

F

G

I

K

L

M

N

O

P

Q

R

S

T

U

V

W