(2)

IDA DOCUMENT D-1002

SOFTWARE REQUIREMENTS SPECIFICATION FOR THE NODE AND VIEW MANAGEMENT TOOL

Clyde G. Roby, Task Leader

David J. Carney
David S. Hough
Richard P. Morton
Jonathan D. Wood

July 1991

Prepared for
Ada Joint Program Office

INSTITUTE FOR DEFENSE ANALYSES
1801 N. Beauregard Street, Alexandria, Virginia 22311-1772

IDA Log No. HQ 91-038806

DEFINITIONS

IDA publishes the following documents to report the results of its work.

Reports

Reports are the most authoritative and most carefully considered products IDA publishes. They normally embody results of major projects which (a) have a direct bearing on decisions affecting major programs, (b) address issues of significant concern to the Executive Branch, the Congress and/or the public, or (c) address issues that have significant economic implications. IDA Reports are reviewed by outside panels of experts to ensure their high quality and relevance to the problems studied, and they are released by the President of IDA.

Group Reports

Group Reports record the findings and results of IDA established working groups and panels composed of senior individuals addressing major issues which otherwise would be the subject of an IDA Report. IDA Group Reports are reviewed by the senior individuals responsible for the project and others as selected by IDA to ensure their high quality and relevance to the problems studied, and are released by the President of IDA.

Papers

Papers, also authoritative and carefully considered products of IDA, address studies that are narrower in scope than those covered in Reports. IDA Papers are reviewed to ensure that they meet the high standards expected of refereed papers in professional journals or formal Agency reports.

Documents

IDA Documents are used for the convenience of the sponsors or the analysts (a) to record substantive work done in quick reaction studies, (b) to record the proceedings of conferences and meetings, (c) to make available preliminary and tentative results of analyses, (d) to record data developed in the course of an investigation, or (e) to forward information that is essentially unanalyzed and unevaluated. The review of IDA Documents is suited to their content and intended use.

The work reported in this document was conducted under contract MDA 903 89 C 0003 for the Department of Defense. The publication of this IDA document does not indicate endorsement by the Department of Defense, nor should the contents be construed as reflecting the official position of that Agency.

This Document is published in order to make available the material it contains for the use and convenience of interested parties. The material has not necessarily been completely evaluated and analyzed, nor subjected to formal IDA review.

REPORT DOCUMENTATION PAGE

Form Approved
OMB No. 0704-0188

1. AGENCY USE ONLY (Leave blank)	2. REPORT DATE	3. REPORT TYPE AND DATES COVERED
	July 1991	Final

4. TITLE AND SUBTITLE
Software Requirements Specification for the Node and View Management Tool

5. FUNDING NUMBERS
MDA 903 89 C 0003

T-D5-496

6. AUTHOR(S)
Clyde G. Roby, David J. Carney, David S. Hough, Richard P. Morton, Jonathan Wood

7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)

Institute for Defense Analyses (IDA)
1801 N. Beauregard Street
Alexandria, VA 22311-1772

8. PERFORMING ORGANIZATION REPORT NUMBER
IDA Document D-1002

9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)
Ada Joint Program Office
The Pentagon, Room 3E114
Washington, DC 20301-3081

10. SPONSORING/MONITORING AGENCY REPORT NUMBER

11. SUPPLEMENTARY NOTES

12a. DISTRIBUTION/AVAILABILITY STATEMENT
Approved for public release, unlimited distribution: ~~25 June 1993~~.

12b. DISTRIBUTION CODE
2A

13. ABSTRACT (Maximum 200 words)

This document specifies the requirements for a Node and View Management (NVM) Tool and is directed to the designers and developers of such a tool. The NVM Tool has been designed and implemented and has been hosted on the NATO Special Working Group (SWG) Ada Programming Support Environment (APSE). The NVM Tool enables tool-writers to create, delete, and modify node and view definitions for the NATO SWG APSE Node Model (i.e., its database). A node is a representation within the Node Model of an entity relevant to the NATO SWG APSE. A view, which is a collection of nodes and their relationships, provides names for certain definitions and their properties. A separate User's Manual has been written for the NVM Tool and provides a more detailed description of the tool.

14. SUBJECT TERMS
Node and View Management Tool; Ada Programming Support Environment (APSE); NATO; Common APSE Interface Set (CAIS); Software Requirements.

15. NUMBER OF PAGES
70

16. PRICE CODE

17. SECURITY CLASSIFICATION OF REPORT	18. SECURITY CLASSIFICATION OF THIS PAGE	19. SECURITY CLASSIFICATION OF ABSTRACT	20. LIMITATION OF ABSTRACT
Unclassified	Unclassified	Unclassified	SAR

IDA DOCUMENT D-1002

SOFTWARE REQUIREMENTS SPECIFICATION FOR THE NODE AND VIEW MANAGEMENT TOOL

Clyde G. Roby, *Task Leader*

David J. Carney
David S. Hough
Richard P. Morton
Jonathan D. Wood

July 1991

INSTITUTE FOR DEFENSE ANALYSES

Contract MDA 903 89 C 0003
Task T-D5-496

Preface

This document specifies the requirements for a Node and View Management (NVM) Tool, which will be hosted on the NATO Special Working Group (SWG) Ada Programming Support Environment (APSE).

This document fulfills the requirement specified in Task Order T-D5-496 (Amendment No. 5), NATO Special Working Group on Ada Programming Support Environments (NATO SWG on APSEs), to "specify, design, and develop a tool hosted on the SWG APSE that will enable tool-writers to create, delete, and modify node and view definitions for the SWG APSE Node Model." This document is directed to the designers and developers of such a tool, giving a list of Specifications for its design.

The Node and View Management Tool was designed and implemented in late 1989. Since then, later versions of the tool have been developed and distributed within the United States. A Users Manual has been written for NVM and gives a much more detailed description of the tool as it exists today.

This document was reviewed by Mr. William Akin, Dr. Randy Garrett, Dr. Richard Ivanetich, Mr. Robert Knapper, Mr. Terry Mayfield, Dr. Richard Wexelblat, and Mr. David Wheeler. The contributions of these reviewers are gratefully acknowledged.

TABLE OF CONTENTS

TABLE OF CONTENTS

TABLE OF CONTENTS

TABLE OF CONTENTS

1. INTRODUCTION

This Software Requirements Specification (SRS) establishes the requirements of the NVM Tool. Its does this by defining the minimal NVM Command Language and by specifying the overall operation of the NVM Tool on a command by command basis. Both the NVM Command Language and the commands described herein must be implemented by the NVM Tool. Additional commands and NVM Command Language constructs may also be implemented by the NVM Tool. However, they must be well documented.

2. SCOPE

2.1 Identification

This Software Requirements Specification establishes the requirements for the Node and View Management Tool (NVM). The NVM Tool (also called simply "NVM" within this document) is intended to be delivered to the NATO Special Working Group (SWG) on Ada Programming Support Environments (APSE) as one of the tools hosted upon it.

2.2 Purpose

The NATO SWG APSE shall support the development and maintenance of Ada application software. The NATO SWG APSE, which consists of a set of integrated tools, will be hosted on a DEC VAX[1]/VMS[2] and ported to a SUN/UNIX[3] computer system. The main purpose of the NVM Tool is to provide users with a capability to create, modify and delete node, attribute, and relationship definitions within a CAIS database. It shall also provide the capability to perform a variety of other functions.

The NVM commands shall provide facilities for invoking programs, for reporting on various states of entities within the Common APSE Interface Set (CAIS) implementation upon which the NVM Tool resides, and for performing CAIS operations.

2.3 Contents of Document

This document contains 6 chapters and 2 appendixes. Chapter 2 addresses the Scope of the document, including its purpose. Chapter 3 identifies applicable government and civil documents. The requirements for the Node and View Management Tool, its programming and design requirements, interface requirements, detailed functional requirements, adaptation requirements, and quality requirements are described in Chapter 4. Chapter 5 addresses qualification

1. VAX is a registered trademark of Digital Equipment Corporation.
2. VMS is a registered trademark of Digital Equipment Corporation.
3. UNIX is a registered trademark of AT&T.

requirements. Delivery of the software code and its documentation is addressed in Chapter 6. Chapter 7 lists acronyms used throughout this document. A sample dialog between a user and the NVM Tool is given in Appendix A. Guidelines for a possible syntax for an NVM Command Language are given in Appendix B.

3. APPLICABLE DOCUMENTS

3.1 Government Documents

The following documents of the exact issue shown form a part of this specification to the extent specified herein. In the event of conflict between the documents referenced herein and the contents of this specification, the contents of this specification shall be considered a superceding requirement.

(1) *Military Standard Defense System Software Development*, DOD-STD-2167A, 29 February 1988.

(2) *Reference Manual for the Ada Programming Language, ANSI/MIL-STD-1815A-1983*, 17 February 1983.

(3) *Military Standard Common APSE Interface Set (CAIS) Revision A*, MIL-STD-1838A, 6 April 1989.

(4) *Requirements for the Ada Programming Support Environment "STONEMAN"*, Department of Defense, February 1980.

3.2 Non-Government Documents

The following documents of the exact issue shown form a part of this specification to the extent specified herein. In the event of conflict between the documents referenced herein and the contents of this specification, the contents of this specification shall be considered a superceding requirement.

(1) *NATO SWG APSE Requirements*, Version 1.0, 21 May 1987.

(2) *Specifications for the Special Working Group Common Ada Programming Support Environment (APSE) Interface Set (CAIS) Implementations, (SWG CAIS Delta Document Version 2.0)*, 8 September 1989.

(3) *VAX/VMS SWG CAIS Users Manual*, Version 5.5, December 7, 1990.

Copies of specifications, standards, drawings, and publications required by suppliers in connection with specified procurement functions should be obtained from the contracting agency or as directed by the contracting officer.

Technical society and technical association specifications and standards are generally available for reference from libraries. They are also distributed among technical groups and the applicable Federal Agencies.

4. REQUIREMENTS

Throughout the remainder of this document, the word "may" appearing in a requirement is defined to mean that the requirement is optional, is of secondary priority, or is a goal.

4.1 Programming Requirements

4.1.1 Programming Language(s)

The programming language for the NVM Tool shall be Ada as defined in ANSI/MIL-STD-1815A.

4.1.2 Compiler/Assembler/Linker

NVM, when integrated with the complete set of SWG APSE tools, shall be compiled using the SWG APSE compiler and shall be linked using the SWG APSE linker. Development of the NVM Tool may be carried out on another validated compiler, with care being taken to ensure that all facilities used are available in the SWG APSE compiler.

4.1.3 Programming Standards

Programming standards should be in accordance with DOD-STD-2167A.

4.1.4 Documentation Requirements

The following disclaimer of warranty and liability shall appear within Ada comments at the very beginning of all Ada language source files and all user documents associated with the NVM Tool:

DISCLAIMER OF WARRANTY AND LIABILITY

THIS IS EXPERIMENTAL PROTOTYPE SOFTWARE. IT IS PROVIDED "AS IS" WITHOUT WARRANTY OR REPRESENTATION OF ANY KIND. THE INSTITUTE FOR DEFENSE ANALYSES (IDA) DOES NOT WARRANT, GUARANTEE, OR MAKE ANY REPRESENTATIONS REGARDING THIS SOFTWARE WITH RESPECT TO CORRECTNESS, ACCURACY, RELIABILITY, MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE, OR OTHERWISE.

USERS ASSUME ALL RISKS IN USING THIS SOFTWARE. NEITHER IDA NOR ANYONE ELSE INVOLVED IN THE CREATION, PRODUCTION, OR DISTRIBUTION OF THIS SOFTWARE SHALL BE LIABLE FOR ANY DAMAGE, INJURY, OR LOSS RESULTING FROM ITS USE, WHETHER SUCH DAMAGE, INJURY, OR LOSS IS CHARACTERIZED AS DIRECT, INDIRECT, CONSEQUENTIAL, INCIDENTAL, SPECIAL, OR OTHERWISE.

4.2 Design Requirements

4.2.1 *Sizing and Timing Requirements*

NVM shall operate within the hardware and software configurations specified by the NATO SWG APSE requirements. The NVM Tool shall share the allocated resources with the other tools in the SWG APSE.

4.2.2 Design Constraints

NVM shall be designed to maximize the portability of the SWG APSE between host systems supporting the SWG CAIS. Thus, the SWG CAIS shall be used extensively by NVM to provide host operating system services. A host operating system service shall not be directly called by NVM if an equivalent service can be provided by the SWG CAIS. NVM may use some of the functions provided by the packages CAIS_ADMINISTRATIVE_SERVICES and CAIS_HOST_SPECIFIC_SERVICES in order to properly execute.

4.3 Interface Requirements

In this section and the next section, several requirements for NVM commands are presented. Although most of the NVM commands do not have an explicit syntax, there are some that do. The NVM Tool shall recognize the syntax of these few commands.

Throughout this section, items that appear in a computer typeface, e.g., `command`, can be either lower or upper case, but must be the same letters. Items that appear in italics, e.g., *parameter* or ***parameter***, must be replaced with an actual corresponding value. Items that appear within curly braces, i.e., { and }, indicate that the items are optional and need not be entered.

(A sample dialog between a user and the NVM Tool is given in Appendix A, "Sample Interactive Session". Command names and syntax are not required to be as shown in this Appendix. Guidelines for a possible syntax for an NVM Command Language are given in Appendix B, "Guideline for NVM Command Language Syntax". These are guidelines only and are not requirements for an NVM Command Language.)

4.3.1 Interface Relationships

NVM shall provide a command that allows it to invoke another process, passing parameters as necessary.

4.3.2 Interface Identification and Documentation

The information pertaining to the CAIS is found in the Common APSE Interface Set (CAIS), MIL-STD-1838A document.

4.3.3 Detailed Interface Requirements

4.3.3.1 NVM to CAIS Interfaces

The SWG CAIS provides the host operating system interface for NVM. NVM may use some of the functions provided by the package CAIS_ADMINISTRATIVE_SERVICES in order to properly execute. There shall be a strong functional cohesion between the SWG CAIS and NVM as one of the functions of NVM shall be to provide nearly all CAIS operations. In addition, all operations that deal with communicating between processes, invoking programs, or obtaining the

status of the SWG APSE, shall require direct access to the CAIS node model. In order to perform its functions, NVM shall utilize, as necessary, CAIS subprograms and types as defined in the CAIS packages.

NVM shall provide commands to manipulate the CAIS nodes, relationships, and attributes.

NVM assumes that each SWG APSE user shall have a unique name, and that this name shall be the key of the `'USER` relation established for the user in the SWG CAIS. This relation shall terminate at a SWG CAIS user node which shall serve as the top level node for all user files.

NVM assumes SWG CAIS group nodes will be established to control access to SWG APSE files and tools. However, these controls are not required for NVM. NVM may provide a means to create and manipulate SWG CAIS groups.

NVM may provide commands to define users and groups. NVM assumes that the process that logs a user onto the SWG APSE will create an executing process from the executable image of NVM for some subset of all users.

Every user who executes NVM may have an initialization file whose pathname is given by `'USER(`*user_name*`)'DOT(LOGIN)`. If the initialization file is present, NVM commands contained in this file shall be interpreted every time that user executes NVM.

Note that several issues regarding the initialization of the CAIS node model at the time of the initialization of the SWG APSE and during the logon time of each user have not been addressed. These operations are considered to be outside the scope of NVM and the CAIS. NVM requires a number of actions before it can operate properly:

(1) The executable image of NVM must be available in a node within the SWG APSE.

(2) NVM must be established as the root process node for some subset of users when they logon to the SWG APSE.

(3) The secondary relationships for `DEVICE` and `USER` need to be established for the NVM process node.

(4) The access rights for NVM must be set properly so that NVM can function as required.

4.4 Detailed Functional Requirements

NVM shall be used to process statements formulated in the NVM Command Language.

Comments, specified in Ada syntax, shall be included in the NVM Command Language. They shall not perform any NVM function but shall be provided to allow for user documentation.

All commands within NVM may be invoked interactively by specifying the command name and parameter values in a common form, similar to the form of an Ada procedure call. When invoked interactively, some parameters for some of the NVM commands may be prompted for and verified by NVM.

NVM shall be tailorable as it shall provide the ability to define aliases for commands and to initialize portions of the environment for a particular user.

When some users logon to the SWG APSE, NVM may be the root process node for them.

The subparagraphs within this section provide the framework for the design of the NVM Tool. When specific syntax of certain commands are given in these subparagraphs, the NVM Tool shall recognize, at a minimum, this syntax.

4.4.1 Initializing and Terminating NVM

For some users, NVM shall be invoked as the top level process when they are logged onto the SWG APSE.

NVM shall look for the existence of a particular file node whose pathname is given by `'USER(`*user_name*`)'DOT(LOGIN)`. If such a file node exists, NVM will read all the lines in that file and execute them as if they had been entered by the user. This allows the user to have the same environment within NVM each time he logs onto the SWG APSE without having to reenter the commands manually.

Once all initialization has occurred, NVM shall begin interpreting user input interactively.

NVM terminates when the interactive user enters the `bye` command or an alias which translates to the `bye` command is entered. If NVM is the top level process, this command shall cause the user to be logged out of the SWG APSE.

4.4.2 Processing Keyboard Input

All keyboard input shall be retrieved and processed by NVM through either the CAIS package CAIS_TEXT_IO or the CAIS package CAIS_TERMINAL_IO[4]. NVM, through the several interfaces to CAIS_TERMINAL_IO, shall recognize three categories of input: immediate signals, editing directives, and NVM commands. Immediate signals shall be those actions that require immediate servicing, such as "abort current command." Editing directives shall be

4. Throughout the remainder of this document, it is assumed that the NVM Tool has been invoked using CAIS_TERMINAL_IO as the means to communicate with the user, unless explicitly documented.

commands (e.g., move cursor left) that are used interactively to modify and correct the NVM commands to be interpreted by NVM. All other input from the keyboard that does not fall into these two categories shall be passed to NVM as part of NVM commands.

Immediate signals shall be passed directly to NVM where the appropriate action shall be performed as soon as possible. Immediate signals shall not be echoed to the terminal device.

Editing directives shall be processed directly by NVM. They shall provide simple character and line oriented functions to aid the user in supplying correct input to NVM. Editing directives shall be associated with keys to ensure that they can be interpreted properly.

NVM commands shall satisfy the input requests to NVM. NVM shall manage a command recall facility which shall provide the user with access to previously stored NVM commands.

4.4.3 Editing Directives

Since the insert mode of operation shall accept NVM commands, editing directives shall not be required for the insertion of characters. The editing directives shall not be echoed to the screen but their effects shall be seen in the change to the program data in the input line, which shall be echoed to the screen. Editing directives shall be associated with keys to ensure that they can be interpreted properly.

NVM shall include at least the following editing directives and key bindings:

CTRL/B	Move cursor left.
CTRL/F	Move cursor right.
CTRL/E	Move cursor to end of line.
CTRL/A	Move cursor to beginning of line.
DEL	Delete character.
CTRL/U	Delete entire line.
CTRL/P	Recall previous NVM command.
CTRL/N	Recall next NVM command.
CR	Indicate end of input.

4.4.3.1 Move cursor left

The current cursor position shall be modified by one column to the left. If the cursor is already at the leftmost position in the input line, then no change shall take place. The current position of the cursor shall be echoed to the screen.

4.4.3.2 Move cursor right

The current cursor position shall be modified by one column to the right. If the cursor is already at the rightmost position in the input line, then no change shall take place. The current position of the cursor shall be echoed to the screen.

4.4.3.3 Move cursor to end of line

The current cursor position shall be modified by making it point to the rightmost position in the input line, i.e., one position past the last non-blank character on the input line. If the current cursor position is already past the last non-blank character on the input line, the cursor position does not change. The current position of the cursor shall be echoed to the screen.

4.4.3.4 Move cursor to beginning of line

The current cursor position shall be modified by making it point to the leftmost position in the input line. If the cursor is already at the leftmost position in the input line, then no change shall take place. The current position of the cursor shall be echoed to the screen.

4.4.3.5 Delete character

The character in the input line at the current cursor position shall be deleted. Every character to the right of the deleted character shall be shifted one position to the left in the input line. The updated contents of the input line shall be echoed to the screen.

4.4.3.6 Indicate end of input

The contents of the input line shall be made available to NVM. The contents of the input line may also be added to the Command History capability for later recall.

4.4.3.7 Recall previous NVM command

The previous NVM command which was entered interactively shall be retrieved via the Command History capability and the current position shall be updated to point to the recalled NVM command. The recalled NVM command shall be placed in the input line and echoed on the screen. If there are no previous entries available via the Command History capability, then no change shall be made to the input line or to the current cursor position.

4.4.3.8 Recall next NVM command

The next NVM command which was entered interactively shall be retrieved via the Command History capability and the current position shall be updated to point to the recalled NVM command. The recalled NVM command shall be placed in the input line and echoed on the screen. If there are no following entries available via the Command History capability, then no change shall be made to the input line or to the current cursor position.

4.4.4 Immediate Signal Processing

All immediate signals shall be handled by NVM as soon as possible after they have been received. Note that because NVM uses the CAIS_TERMINAL_IO package for its communication with the user, immediate signals are checked only when input or output is performed via CAIS_TERMINAL_IO. Immediate signal functions shall not be echoed to the terminal. The following is a list of immediate signal functions:

(1) Abort current NVM command.

(2) Abort the NVM program.

These functions must be immediate signals but there are perhaps other functions which may qualify for immediate processing. The following set of commands consists of the immediate signal commands that shall be processed by NVM and their complementary functions.

4.4.4.1 Abort current NVM command

When this signal is detected by NVM, an Ada exception shall be raised. This exception may be detected in the course of the execution of any NVM commands so that any cleanup operations for those commands may be performed.

4.4.4.2 Abort the NVM program

In the current implementation of the NATO SWG APSE CAIS, entering the keyboard equivalent of end-of-file (CTRL/Z for DEC VAX/VMS) gives an indication of END_OF_FILE for the CAIS_TERMINAL_IO package. NVM should recognize this condition and raise an appropriate Ada exception. All exception handlers that recognize this exception should reraise the exception so that the main NVM processing loop can recognize the condition and exit the NVM Tool.

4.4.5 Tailoring NVM

Several features shall be available to enable tailoring of the NVM Tool in the SWG APSE environment:

(1) Command name aliasing shall allow the user to rename an existing NVM command.

(2) Pathname aliasing shall allow the user to name a pathname (e.g., so that a fewer number of characters may be used for that pathname in NVM Command Lines).

(3) Each user shall have his own set of NVM commands that shall be interpreted when that user logs in to NVM. The pathname of the file containing these commands is `'USER(`*user_name*`)'DOT(LOGIN)`.

These features are described below.

4.4.5.1 Command name aliasing

If certain NVM commands are not to the liking of the user, then the command names may be aliased. It shall be possible to alias all NVM commands, except for the **bye** command (the **bye** command allows the user to exit from the NVM Tool). The syntax of the command to perform command name aliasing shall be:

alias [from=>]*user_name* [to=>]*existing_name*

Once the *user_name* and *existing_name* parameters have been verified, the alias shall replace any alias currently defined with the same *user_name*. This ensures that a command name alias shall refer only to one command at a time. An error message shall be displayed if an attempt is made to redefine, via an alias, the **bye** command.

4.4.5.2 Pathname aliasing

The user may also alias parameters that are pathnames. At least two parameters must be given with this command, an existing pathname and the user's new pathname alias. Once these parameters have been verified, the pathname alias shall replace any pathname alias currently defined with the same existing pathname. This ensures that a pathname alias shall refer only to one pathname at a time.

4.4.6 Parsing NVM Commands

Comments, specified in Ada syntax, shall be included in the NVM Command Language. They shall not perform any NVM function but shall be provided to allow for user documentation.

Command name aliases and pathname aliases shall be dealiased such that the command or the pathname shall replace the respective alias in the NVM Command Line before it is parsed. Once aliases are processed, the NVM Command Line shall be parsed.

NVM commands shall be parsed before they can be processed.

4.4.7 Prompting for parameters

To ensure that NVM is easy to use, the ability to prompt for parameters shall be provided. NVM may prompt the user for further program data, i.e., additional data needed for NVM commands, if an insufficient number of parameters is supplied for a command. NVM shall supply default values for parameters which, in practice, almost always have the same value, e.g., attributes, intents, etc. If values are provided for all parameters which do not have default values, then the number of parameters supplied shall be considered sufficient.

If a command is specified with too few parameters, an error shall be displayed. The user should then enter values for all essential parameters and modify any of the default values for parameters that are not suitable. Once the user is finished providing input for the NVM command, NVM shall interpret the command.

4.4.8 Initiating processes

NVM shall provide three facilities for initiating processes.

(1) The first type of initiated process shall be referred to as a *spawned process*; it shares the same environment as the NVM Tool. NVM and the spawned process operate independently of each other.

(2) The second type of initiated process shall be referred to as an *invoked process*; it shares the same environment as the NVM Tool. NVM must wait until the invoked process completes before it regains control of the environment.

(3) The third type of initiated process shall be referred to as a *detached process* or a *job*. NVM and the new job shall then be able to execute concurrently and independently. Since the created job shall form its own process tree, it shall not be affected by the user logging off.

4.4.8.1 Starting spawned processes

NVM shall have commands to spawn processes and start them.

4.4.8.2 Starting invoked processes

NVM shall have commands to invoke processes and start them.

4.4.8.3 Starting detached processes

NVM shall have commands to create jobs and start them.

4.4.9 Providing Parameter Lists for processes

The input parameters provided for a process shall be formatted by NVM and then stored in the predefined PARAMETERS attribute of the invoked process. It shall be the responsibility of the invoked program to extract the parameter list from the PARAMETERS attribute associated with its process node and to use CAIS facilities to extract the values in their appropriate forms.

Output parameters shall be stored by the invoked program in the predefined RESULTS attribute of the associated process node for retrieval by another NVM command. As well as retrieving the results, NVM shall obtain the status of the process. The process status shall be stored in the predefined CURRENT_STATUS attribute.

Problems encountered by NVM while storing parameters or results shall be reported to the user.

4.4.10 Interpreting CAIS pathnames

NVM shall provide the interactive user with direct access to many CAIS facilities. As a result, many of the commands in NVM shall require the specification of a pathname. Pathnames shall be used to identify nodes in the SWG APSE. These pathnames shall take the same form as CAIS pathnames, consisting of relation names and relationship key designators. Refer to the CAIS specification document for a complete description of CAIS pathnames.

Pathnames shall be considered to be full pathnames if they start with an apostrophe (or *tick*); otherwise, they shall be abbreviated pathnames. There may be several forms of abbreviated pathnames. NVM shall support at least the form of an abbreviated pathname for which the full pathname may be obtained by prefixing it with `'CURRENT_NODE'` so that the pathname given by `DOT(SMITH)` is equivalent to the pathname `'CURRENT_NODE'DOT(SMITH)` (or `'CURRENT_NODE.SMITH`). Wherever pathnames are required, both of these forms of pathnames shall be acceptable.

Apart from simply referring to a single pathname, it shall be possible to have wildcard characters in the name to allow for reference to several nodes, relationships, or attributes. The character "*" shall match zero or more characters within a relation name or relationship key designator. The character "?" shall match any single character. Wildcard characters may be used in pathnames if it is explicitly noted in the description of the NVM command; otherwise, they shall not be allowed.

4.4.11 Manipulating the SWG APSE Database

In order to use the SWG APSE as a development system, the user requires the ability to manipulate the CAIS nodes, including all the definition and view nodes. In the following sections, requirements for manipulating the various nodes, relationships, and attributes of the SWG APSE database are described.

4.4.12 Nodes, Relationships, and Attributes: Creation and Deletion

4.4.12.1 Create Structural Node

This command shall create a Structural node in the SWG APSE. At least two parameters must be given with this command, the relation (which may have a default) and the relationship key of the relationship to the new Structural node to be created. Any problems encountered while creating the node shall be reported to the user.

4.4.12.2 Create Secondary Relationship

This command shall create a secondary relationship emanating from one node and targeted to another node. At least three parameters must be given with this command: the name of the secondary relationship, the identification of the source node (which may have a default), and the identification of the target node. For those relationships that require a key, a fourth parameter must also be given with this command: the value of that key.

4.4.12.3 Create Bidirectional Secondary Relationship

This command shall create a bidirectional secondary relationship emanating from one node and targeted to another node. At least three parameters must be given with this command: the name of the secondary relationship, the identification of the source node (which may have a default), and the identification of the target node. For those bidirectional relationships that require a key, two additional parameters may also be given with this command: the values of the keys for the secondary relationship and the reverse relationship. Any problems encountered while creating the bidirectional secondary relationship shall be reported to the user.

4.4.12.4 Delete Secondary Relationship

This command shall delete a secondary relationship emanating from a node in the SWG APSE. At least two parameters must be given with this command, the name of the secondary relation to be deleted from a node, and the identification of that node (which may have a default) in the SWG APSE. Any problems encountered while deleting the secondary relationship shall be reported to the user.

4.4.12.5 Set Node Attribute

This command shall set (or change) the value of an attribute on a node in the SWG APSE. At least three parameters must be given with this command, the identification of the node whose attribute is to be changed (which may have a default), the name of the node attribute, and the new value to be set for the node attribute. Any problems encountered while setting the node attribute shall be reported to the user.

4.4.12.6 Set Path Attribute

This command shall set (or change) the value of an attribute on a relationship in the SWG APSE. If the attribute does not exist, it is first created. At least three parameters must be given with this command, the name of the relationship whose attribute is to be changed, the name of the path attribute on that relationship, and the new value to be set for the path attribute. Any problems encountered while setting the path attribute shall be reported to the user.

4.4.12.7 Copy Node

**`copyn [from=>]`*`from_pathname`*` [to=>]`*`to_pathname`*

This command shall create a copy of a single Structural or Secondary Storage node. At least two pathnames are required for this command. The *from_pathname* shall identify the node that is to be copied. The *to_pathname* shall specify the name of the new node that is to be a copy of the first node; this pathname must not specify an existing node. Any problems encountered while copying the node shall be reported to the user.

4.4.12.8 Copy Tree

**`copyt [from=>]`*`from_pathname`*` [to=>]`*`to_pathname`*

This command shall create a copy of a node tree. At least two pathnames are required for this command. The *from_pathname* shall identify the root node of the tree that is to be copied. The *to_pathname* shall specify the name of the new node that is to be a copy of the root node and under which a copy of all the nodes in the node tree are to be placed; this pathname must not specify an existing node. Any problems encountered while copying the tree shall be reported to the user.

4.4.12.9 Delete Node

```
deln {node=>}node_pathname
```

This command shall delete the primary relationship to a node in the SWG APSE; the node becomes unobtainable. This command shall require at least one parameter, *node_pathname*, which shall identify the pathname to an existing node in the SWG APSE. Any problems encountered while deleting the Structural node shall be reported to the user.

4.4.12.10 Delete Tree

```
delt {node=>}node_pathname
```

This command shall delete a node tree from the SWG APSE. This command shall require at least one parameter, *node_pathname*, which shall identify the pathname to an existing node tree in the SWG APSE. Any problems encountered while deleting the node tree shall be reported to the user.

4.4.12.11 Move Node

```
move {from=>}from_pathname {to=>}to_pathname
```

This command shall move a single Structural or Secondary Storage node. At least two pathnames are required for this command. The *from_pathname* shall identify the node that is to be moved (or renamed). The *to_pathname* shall specify the pathname of the node after it has been moved. Any problems encountered while moving (renaming) the node tree shall be reported to the user.

4.4.13 View Operations

The following commands shall be supported by the NVM Tool. The number of parameters and the parameters themselves are not detailed in this section. However, a list of appropriate functions is included here for completeness. The NVM Tool shall have commands which shall:

(1) Adopt a View,

(2) Unadopt a View,

(3) Create a View, and

(4) Delete a View.

Any problems encountered while executing any of these commands shall be reported to the user.

4.4.14 Definition Node Operations

The following commands shall be supported by the NVM Tool. The number of parameters and the parameters themselves are not detailed in this section. However, a list of appropriate functions is included here for completeness. The NVM Tool shall have commands which shall:

(1) Create an Attribute Definition (AD) node,

(2) Create an Attribute Kind Definition (AKD) node,

(3) Create a Node Kind Definition (NKD) node,

(4) Create a Node Kind Interpretation (NKI) node for an existing node kind,

(5) Create a Node Kind Specialization by designating an existing node kind to be a specialization of another existing node kind,

(6) Create a Relation Specialization by designating one existing relation to be a specialization of another existing relation,

(7) Create a Unidirectional Secondary Relation Definition node and its associated Relation Interpretation node,

(8) Create a Bidirectional Secondary Relation Definition (RD) node represented by a single RD node (for bidirectional relationships whose two unidirectional components are semantically identical) and the associated Relation Interpretation node,

(9) Create a Bidirectional Relation Definition pair of nodes, i.e., create a Bidirectional Relation Definition represented by two distinct Bidirectional Relation Definition nodes (for bidirectional relationships whose two unidirectional components are semantically distinct) and their associated Relation Interpretation nodes; these Relation Definitions can be designated either as primary or secondary Relation Definitions,

(10) Create a Relation Interpretation (RI) node for an existing Relation Definition node,

(11) Define an Emanating Relation by creating a secondary EMANATES relationship from an NKD node to a Relation Definition node, thus indicating that relationships of the Relation defined by the RD node can emanate from nodes (instances) of the specified node kind definition,

(12) Define a Terminating Relation by creating a secondary TERMINATES relationship from an NKD node to a Relation Definition node, thus indicating that relationships of the Relation defined by the RD node can terminate at nodes (instances) of the specified node kind definition,

(13) Define an Attribute for either a node kind (NKD) or a relation kind (RD),

(14) Delete an Attribute Definition, i.e., delete a secondary relationship which defines a local name for an attribute definition and optionally delete the corresponding Attribute Definition node,

(15) Delete an Attribute Kind Definition, i.e., delete a secondary relationship which defines a local name for an attribute kind definition and optionally delete the corresponding Attribute Kind Definition node,

(16) Delete a Node Kind Interpretation node from a View and optionally the corresponding Node Kind Definition node,

(17) Delete a Node Kind Specialization, i.e., sever a specialization association between two nodes,

(18) Delete a Relation Specialization, i.e., sever a specialization association between two relations,

(19) Delete a Relation Interpretation and Definition, i.e., delete a Relation Interpretation node for either a unidirectional or bidirectional relation from a View and optionally the corresponding Relation Definition node for the unidirectional relation,

(20) Delete a Bidirectional Relation Interpretation and Definition, i.e., delete a Relation Interpretation node for a bidirectional relation from a View and optionally the corresponding Bidirectional Relation Definition node,

(21) Delete a tree of definitions,

(22) Delete an Emanating Relation, i.e., delete a secondary relationship which defines a local name for a relation and optionally delete a secondary EMANATES relationship,

(23) Delete a Terminating Relation, i.e., delete a secondary TERMINATES relationship from a Node Kind Definition node to a Relation Definition node,

(24) Delete an Attribute from an NKD or an RD, i.e., delete from a view a secondary ATTRIBUTE_NAME relationship which defines a local name for an attribute and optionally deletes a secondary ATTRIBUTE relationship emanating from the NKD or RD node and terminating at the AD node,

(25) Set a Definition Node Attribute value, i.e., set the value of a CAIS-controlled definition node attribute,

(26) Convert a node, i.e., change a node from an instance of one node kind to be an instance of another node kind, and

(27) Display all primary and secondary relationships emanating from a specified Definition node and display all node and relation attributes.

Any problems encountered while executing any of these commands shall be reported to the user.

4.4.15 File Node Operations

The commands described in the following subsections shall be supported.

4.4.15.1 Create File Node

This command shall create a File node in the SWG APSE. At least two parameters must be given with this command, the relation (which may have a default) and the relationship key of the relationship to the new File node to be created. Any problems encountered while creating the node shall be reported to the user.

4.4.15.2 Create Executable Image File Node

This command shall create an Executable Image File node in the SWG APSE. At least two parameters must be given with this command, the relation (which may have a default) and the relationship key of the relationship to the new Executable Image File node to be created. Any problems encountered while creating the node shall be reported to the user.

4.4.15.3 Delete Node

This command shall delete the primary relationship to a node in the SWG APSE. At least one parameter must be given with this command, the pathname to the existing node to be deleted. Any problems encountered while deleting the node shall be reported to the user.

4.4.15.4 Export File Contents

```
expfc [from=>]CAIS_file_node_pathname [to=>]host_filename
```

This command shall export the contents of a File node from the SWG APSE to the host file system. There shall be at least two parameters to this command. The *CAIS_file_node_pathname* shall be the pathname of the File node in the SWG APSE that contains the file that is to be exported to the host file system. The *host_filename* shall be a valid host file system filename. Any problems encountered while exporting the file node contents shall be reported to the user.

4.4.15.5 Import file contents

```
impfc [from=>]host_filename [to=>]CAIS_file_node_pathname
```

This command shall import the contents of a file from the host file system to the SWG APSE. There shall be at least two parameters to this command. The *host_filename* shall be a valid host file system filename of the file to be imported into the SWG APSE. The *CAIS_file_node_pathname* is the pathname of the File node in the SWG APSE that the file is to be imported to. Any problems encountered while importing the file node contents shall be reported to the user.

4.4.15.6 List File Contents

This command shall list (display) the contents of a File node (Secondary Storage Text). There shall be at least one parameter to this command, the pathname of the File node whose contents are to be listed. If the File node can be opened, then the contents of the File node are listed (displayed) on the user's terminal. Any problems encountered while displaying the file node contents shall be reported to the user.

4.4.15.7 Write File Contents from Terminal Input

This command shall write the contents of a File node from what the user enters from the terminal. There shall be at least one parameter to this command, the pathname of the File node whose contents are to be written to. Any problems encountered while writing the file node contents shall be reported to the user.

4.4.16 Inspection Operations

The commands described in the following subsections shall be supported.

4.4.16.1 Display whether or not a Node Is Obtainable

This command shall determine if a node is obtainable or accessible. Either `TRUE` or `FALSE` shall be displayed at the user's terminal. There shall be at least one parameter to this command, the pathname of the node to be queried for Obtainability. Any problems encountered while determining node obtainability shall be reported to the user.

4.4.16.2 Display whether or not Two Pathnames Identify the Same Node

This command shall compare two pathnames. If the two pathnames refer to the same CAIS node, `TRUE` shall be displayed. Otherwise `FALSE` shall be displayed. There shall be two parameters for this command, each shall be a pathname of a node (which may have a default) in the SWG APSE identifying nodes that will be queried for sameness. Any problems encountered while determining whether two pathnames identify the same node shall be reported to the user.

4.4.16.3 List Current Node

This command shall list (display) the full pathname of the node that is the current node for the user on the user's terminal. Any problems encountered while displaying the pathname shall be reported to the user.

4.4.16.4 List File Nodes

This command shall list (display) all File nodes that are reachable from a given node in the SWG APSE. There shall be at least one parameter for this command, the pathname identifying the node (which may have a default) from which all File nodes that are reachable will be displayed to the user. Any problems encountered while displaying the file nodes shall be reported to the user.

4.4.16.5 List Nodes

This command shall list (display) all primary and secondary relationships emanating from a specified node and display all node and relation attributes. There shall be at least one parameter required for this command, the identification of a node (which may have a default) for which all this information is to be displayed. Any problems encountered while displaying the relationships shall be reported to the user.

4.4.16.6 List Node Attributes

This command shall list (display) all the node attributes for a node in the SWG APSE. There shall be at least one parameter for this command, the pathname of the node (which may have a default) whose attributes are to be displayed to the user's terminal. Any problems encountered while displaying the node attributes shall be reported to the user.

4.4.16.7 List Path Attributes

This command shall list (display) all the path attributes for a relationship in the SWG APSE. There shall be at least two parameters for this command, the name of a relation and the name of the key of the relationships whose attributes are to be displayed to the user's terminal. Any problems encountered while displaying the path attributes shall be reported to the user.

4.4.16.8 List Process Nodes

This command shall list (display) all Process nodes that are reachable from a given node in the SWG APSE. There shall be at least one parameter for this command, the pathname identifying the node (which may have a default) from which all Process nodes that are reachable will be displayed to the user. Any problems encountered while displaying the Process nodes shall be reported to the user.

4.4.16.9 List Primary Relationships

This command shall list (display) all the primary relationships emanating from a node in the SWG APSE. There shall be at least one parameter for this command, the pathname of a node (which may have a default) whose primary relationships are to be displayed to the user's terminal. Any problems encountered while displaying the primary relationships shall be reported to the user.

4.4.16.10 List Relationships

This command shall list (display) all the primary and secondary relationships emanating from a node in the SWG APSE. There shall be at least one parameter for this command, the identification of a node (which may have a default) whose primary and secondary relationships are to be displayed to the user's terminal. Any problems encountered while displaying the relationships shall be reported to the user.

4.4.16.11 List Structural Nodes

This command shall list (display) all Structural nodes that are reachable from a given node in the SWG APSE. There shall be at least one parameter for this command, the pathname identifying the node (which may have a default) from which all Structural nodes that are reachable will be displayed to the user. Any problems encountered while displaying the Structural nodes shall be reported to the user.

4.4.16.12 List Secondary Relationships

This command shall list (display) all the secondary relationships emanating from a node in the SWG APSE. There shall be at least one parameter for this command, the pathname of a node (which may have a default) whose secondary relationships are to be displayed to the user's terminal. Any problems encountered while displaying the secondary relationships shall be reported to the user.

4.4.16.13 List Spanning Tree

This command shall list (display) the Spanning Tree of a node in the SWG APSE. The Spanning Tree shall contain all the nodes reachable from the given node by traversing all of its primary relationships recursively. There shall be at least one parameter for this command, the pathname of a node (which may have a default) whose Spanning Tree is to be displayed to the user's terminal. Any problems encountered while displaying the spanning tree shall be reported to the user.

4.4.16.14 Set Current Node

This command shall define the Current Node for the user. All abbreviated pathnames, following successful execution of this command, shall be considered to be relative to the specified Current Node. At least one parameter must be given with this command, the pathname of the node that is to become the Current Node for the user. If the pathname is valid, then the specified node shall become the Current Node. Any problems encountered during the execution of this command shall be reported to the user and this command shall have no effect.

4.4.17 Process Control Commands

In the following list of commands that shall be supported by the NVM Tool, several commands create Process nodes. After the particular process is terminated, NVM shall automatically delete the Process node for that process, by default. There shall be a way that the user can override this automatic deletion capability for each of these commands.

4.4.17.1 Activate And Await Process Termination

This command shall activate a process in the SWG APSE. There shall be at least one parameter, the pathname of the Process node whose process is to be activated and whose process termination is to be waited upon. Any problems encountered while activating the process or awaiting for process termination shall be reported to the user.

4.4.17.2 Activate Process

This command shall activate a process in the SWG APSE. There shall be at least one parameter, the pathname of the Process node whose process is to be activated. Any problems encountered while activating the process shall be reported to the user.

4.4.17.3 Abort Process

abort [process_node=>]*process_pathname*

This command shall abort a process in the SWG APSE. There shall be at least one parameter, *process_pathname* is the pathname of the Process node whose process is to be aborted. Any problems encountered while aborting the process shall be reported to the user.

4.4.17.4 Await Process Termination

This command shall await the termination of a process in the SWG APSE. There shall be at least one parameter, the pathname of the Process node whose process termination is to be waited upon. Any problems encountered while awaiting process termination shall be reported to the user.

4.4.17.5 Append Results

This command shall append a value to the RESULTS attribute of the NVM Tool's Process node. There shall be at least one parameter, the string value (which may be the null string) that is appended to the NVM Tool's Process node's RESULTS attribute. Each time that this command is given, an additional string value shall be added to the RESULTS attribute of the NVM Tool's Process node. Any problems encountered while appending results shall be reported to the user.

4.4.17.6 Create Job

This command shall create and activate a new job in the SWG APSE. Creation of a new job implies creation of a new Job node (Root Process node) in the SWG APSE. There shall be at least one parameter for this command, the pathname of the Executable Image File node from which to get the program that will be the new job. Any problems encountered while creating the job shall be reported to the user.

4.4.17.7 Create Job Node

This command shall create a Job node in the SWG APSE, but shall not activate the job. There shall be at least one parameter for this command, the pathname of the Executable Image File node from which to get the program that will loaded into the new Job node. Any problems encountered while creating the job node shall be reported to the user.

4.4.17.8 Create Process Node

This command shall create a Process node in the SWG APSE, but shall not activate the process. There shall be at least one parameter for this command, the pathname of the Executable Image File node from which to get the program that will loaded into the new Process node. Any problems encountered while creating the process node shall be reported to the user.

4.4.17.9 List Current Status Information of a Process

This command shall list (display) the values of these attributes which appear on a Process node: TIME_STARTED, TIME_FINISHED, MACHINE_TIME, IO_UNIT_COUNT, OPEN_HANDLE_COUNT, and CURRENT_STATUS. There shall be at least one parameter for this command, the pathname of a Process node (which may have a default) in the SWG APSE. The value of the CURRENT_STATUS attribute displayed shall be one of the following: `CREATED`, `READY`, `SUSPENDED`, `COMPLETED_ABNORMALLY`, `COMPLETED_NORMALLY`, `ABORTED`, or `TERMINATED`.

The format of the value of the MACHINE_TIME attribute shall be

```
ss.mmm
```

where `ss` is the value of seconds and `mmm` is the value of milliseconds. The format of the values of the TIME_FINISHED and TIME_STARTED attributes shall be

```
yr/mo/da hh:mm:ss.sss
```

where `yr` is the year, `mo` is the month, `da` is the day, `hh` is the hour, `mm` is the minutes, and `ss.sss` is the seconds expressed to three decimal places. Any problems encountered while displaying the current status of the process shall be reported to the user.

4.4.17.10 List Priorities

This command shall list (display), on the user's terminal, the default process priority of the NVM Tool, the highest user priority of a process in the SWG APSE, and the lowest user priority of a process in the SWG APSE. Any problems encountered while displaying the priorities of the process shall be reported to the user.

4.4.17.11 Set Priority

This command shall set the (default) priority of the current (enclosing) process in the SWG APSE. There shall be at least one parameter for this command, the value of the priority. Any problems encountered while setting the priority of the process shall be reported to the user.

4.4.17.12 Delete Job

This command shall delete a Job node in the SWG APSE. There shall be at least one parameter for this command, the pathname of a Job node to be deleted. Any problems encountered while deleting the job shall be reported to the user.

4.4.17.13 List the Parameters of a Process

This command shall list (display) the value of the PARAMETERS attribute of the NVM Tool's Process node on the user's terminal. Any problems encountered while displaying this value shall be reported to the user.

4.4.17.14 List the Results of a Process

This command shall list (display) the value of the RESULTS attribute of a Process node in the SWG APSE. There shall be at least one parameter for this command, the pathname of a Process node (which may have a default) whose RESULTS attribute is to be displayed on the user's terminal. Any problems encountered while displaying this value shall be reported to the user.

4.4.17.15 Invoke Process

`inv [file_node=>]`*`executable_image_pathname`*

This command shall suspend the NVM Tool and invoke another process in the SWG APSE. There shall be at least one parameter for this command, *executable_image_pathname* is the pathname of an Executable Image File node. NVM shall no longer have control of the user's terminal once NVM invokes the other process. When the invoked process terminates, NVM shall once again gain control and notify the user of the pathname, results, and current status of the terminated process. When NVM invokes the other process, it shall do so in a manner that the Process node shall be deleted automatically upon termination of that process, unless the user has indicated otherwise. If the Process node is not deleted, the user can examine attributes on that Process node. Any problems encountered while invoking the process shall be reported to the user.

4.4.17.16 Resume And Await Termination of a process

This command shall resume and await the termination of a process in the SWG APSE. There shall be at least one parameter, the pathname of the Process node whose process is to be resumed and whose termination is to be waited upon. Any problems encountered while resuming the process or awaiting for process termination shall be reported to the user.

4.4.17.17 Resume Process

```
res {process_node=>}process_pathname
```

This command shall resume a process or process tree in the SWG APSE. There shall be at least one parameter, *process_pathname* is the pathname of the Process node whose process is to be resumed, or, the pathname of the Process node that is the root of the process tree to be resumed. Any problems encountered while resuming the process shall be reported to the user.

4.4.17.18 Spawn Process

```
spawn {file_node=>}executable_image_pathname
```

This command shall spawn another process in the SWG APSE. There shall be at least one parameter for this command, *executable_image_pathname* is the pathname of an Executable Image File node. NVM shall continue to have control of the user's terminal once NVM spawns the other process. When NVM spawns the other process, it shall do so in a manner that the Process node shall be deleted automatically upon termination of that process, unless the user has indicated otherwise. If the Process node is not deleted, the user can examine attributes on that Process node. Any problems encountered while spawning the process shall be reported to the user.

4.4.17.19 Suspend Process

```
susp {process_node=>}process_pathname
```

This command shall suspend a process or process tree in the SWG APSE. There shall be at least one parameter, *process_pathname* is the pathname of the Process node whose process is to be suspended. Any problems encountered while suspending the process shall be reported to the user.

4.4.17.20 Write Results

This command shall write a value to the RESULTS attribute of the NVM Tool's Process node. There shall be at least one parameter, the string value (which may be a null string) that is written to the NVM Tool's Process node's RESULTS attribute. Each time that this command is given, the previous value of the RESULTS attribute is overwritten. Any problems encountered while writing results shall be reported to the user.

4.4.18 Access Control Commands

The commands described in the following subsections shall be supported.

4.4.18.1 Adopt Role

This command shall adopt a role in the SWG APSE. There shall be at least two parameters, a pathname of the group node to be adopted and a key (which may default to the empty key) of a secondary ADOPTED_ROLE relationship that will be created from the NVM Tool to that Group node. Any problems encountered while adopting the role shall be reported to the user.

4.4.18.2 Deny Access

This command shall create a DENY relationship from a node to a Group node. There shall be at least two parameters, each specifying a pathname to the aforementioned nodes. Any problems encountered while creating the relationship shall be reported to the user.

4.4.18.3 Delete Deny Relationship

This command shall delete a DENY relationship from a node to a Group node. There shall be at least two parameters, the pathname of the node from which the DENY relationship emanates and the pathname of the node to which the DENY relationship terminates. Any problems encountered while deleting the relationship shall be reported to the user

4.4.18.4 Delete Granted Rights

This command shall delete an ACCESS relationship from a node to a Group node. There shall be at least two parameters, the pathname of the node from which the ACCESS relationship emanates and the pathname of the node to which the ACCESS relationship terminates. Any problems encountered while deleting the relationship shall be reported to the user

4.4.18.5 Get Granted Rights

This command shall obtain granted rights. There shall be at least two parameters, the pathname of the node (which may have a default) which has granted rights and the pathname of the group node (which also may have a default) to which the rights have been granted. Any problems encountered while obtaining the granted rights shall be reported to the user

4.4.18.6 Is Approved

This command shall display an indication whether or not the NVM Tool as a subject has an approved access right to a node as an object in the SWG APSE and has not been denied access to that node. Either **TRUE** or **FALSE** shall be displayed at the user's terminal. Any problems encountered while determining this shall be reported to the user

4.4.18.7 Set Granted Rights

This command shall create or modify an ACCESS relationship. There shall be at least three parameters, the pathname of the node (which may have a default) from which access control information is to be set, the pathname of the group node (which may have a default) to which access information is to be set, and the list of the access rights to be granted. Any problems encountered while creating or modifying the relationship shall be reported to the user

4.4.18.8 Unadopt Role

This command shall unadopt a role in the SWG APSE. There shall be at least one parameter, the key of the ADOPTED_ROLE relationship to be deleted. Any problems encountered while unadopting the role shall be reported to the user

4.4.19 NVM Utility Commands

The commands described in the following subsections shall be supported.

4.4.19.1 Providing help

The help command shall provide the user with on-line help information.

help [**from=>**]*NVM_command*

If the above form of the **help** command is given, then help information for *NVM_command* shall be displayed.

help

If this form of the **help** command is given, then the user will be given the opportunity to choose from one of the several NVM commands or groups of commands available and help information for that particular command shall be displayed.

In either case, the **help** command shall place the user into an interactive mode while the help text is presented to him[5]. The help facility shall make use of the CAIS_TERMINAL_IO facilities in displaying the information for the user. A new Screen shall be opened for the help information. This Screen shall be used until the user no longer requires the help information, then the Screen shall be removed from the terminal display.

The help text shall contain information on all the commands in the NVM Tool and general help about NVM itself.

4.4.19.2 Exiting from NVM

bye

This command shall terminate the NVM Tool. NVM shall not allow the **bye** command to be redefined as an alias. However, other user-defined commands, e.g., **logout**, can be aliased to the **bye** command. When the **bye** command is processed, the NVM Tool terminates, thus terminating any subprocesses. If NVM is the top level process, then the SWG APSE session shall

---

5. Only with the CAIS_TERMINAL_IO implementation

be terminated and control shall be returned to the host operating system or the calling environment, in general, which could be another instance of the NVM Tool or another CAIS process capable of invoking processes.

4.4.19.3 Display date

```
date
```

This command shall display the current date from the SWG APSE. The date shall be retrieved via the CAIS_CALENDAR package and formatted for display. The format of the date value shall be

```
month day, year
```

where **month** is the name of the month spelled out, **day** is the day of the month, and **year** is the current year.

4.4.19.4 Display time

```
time
```

This command shall display the current time from the SWG APSE. The time shall be retrieved via the CAIS_CALENDAR package and formatted for display. The format of the time value shall be

```
hh:mm:ss xx
```

where **hh** is the hour, **mm** is the minutes, **ss** is the seconds, and **xx** is either **AM** or **PM**.

4.4.19.5 Define Command Name Alias

```
alias [from=>]user_name [to=>]existing_name
```

This command shall define a command name alias. Once the *user_name* and *existing_name* parameters have been verified, the alias shall replace any alias currently defined with the same *user_name*. This ensures that a command name alias shall refer only to one command at a time. An error message shall be displayed if an attempt is made to redefine, via an alias, the **bye** command.

4.4.19.6 Define Pathname Alias

The user may also alias parameters that are pathnames. At least two parameters must be given with this command, an existing pathname and the user's new pathname alias. Once these parameters have been verified, the pathname alias shall replace any pathname alias currently defined with the same existing pathname. This ensures that a pathname alias shall refer only to one pathname at a time.

4.4.19.7 Delete Command Name Alias

```
unalias [from=>]user_name
```

This command shall delete the command name alias given by the *user_name* parameter.

4.4.19.8 Delete Pathname Alias

The user may also delete pathname aliases. At least one parameter must be given with this command, a pathname alias. Once the parameter has been verified, the given pathname alias shall no longer be in effect.

4.4.19.9 Display Command Name Aliases

```
alias [from=>]user_name
```

This form of the command shall display the command name alias and the expanded form of the alias identified by *user_name* on the user's terminal.

```
alias
```

This form of the command shall display all command name aliases and their expanded forms on the user's terminal.

4.4.19.10 Display Pathname Aliases

The user may also display a particular pathname alias and its expanded form. The user may also display all pathname aliases and their expanded forms on the user's terminal.

4.4.19.11 Execute Host Command

```
host [param1=>]string1 [param2=>]string2
```

This command shall pass the parameters *string1* and *string2*, concatenated together as one string, and which contains a host command, to the host system.

4.4.19.12 Execute CAIS System Administration Tool

```
sysadmin
```

This command shall invoke the CAIS System Administration Tool, located in the Executable Image File node identified by the pathname given as `'USER(TOOLS).SYSTEM_ADMINISTRATION_TOOL`.

The NVM Tool may, within NVM, perform functions defined in the CAIS System Administration Tool.

4.5 Adaptation Requirements

4.5.1 System Environment

For the proper functioning of the NVM Tool, several environment tables must be established in File nodes. The tables shall be defined for each installation of the SWG APSE. These include:

(1) Help files.

(2) Function key definition file.

4.5.2 System Parameters

NVM shall execute on the SWG APSE. The SWG APSE will be hosted on a DEC VAX/VMS computer system. The initial operating system version shall be 5.1 or later. The SWG APSE shall also be rehosted on a SUN/UNIX system. The initial operating system version shall be SunOS 4.0 or later. Updates to the operating system(s) and the CAIS implementation(s) shall be transparent to the NVM user.

4.5.3 System Capacities

The CAIS implementation shall be responsible for determining the usage of primary storage (memory) versus secondary storage (disk space) for the node model.

4.6 Quality Requirements

4.6.1 Correctness Requirements

NVM shall be developed in accordance with the requirements detailed in this Software Requirements Specification.

4.6.2 Reliability Requirements

NVM shall be a reliable tool in the SWG APSE. The tolerance for external software errors shall be less than one percent. This tolerance shall be based upon the number of errors detected in NVM versus the number of NVM commands performed. Failure of NVM shall not affect the integrity of the underlying SWG APSE database. At most, failure shall result in the loss of the current working environment.

To ensure reliability of the tool, the following attributes of software shall be considered in the design, development, and quality assurance phases of the NVM implementation:

(1) Accuracy - NVM must function as specified.

(2) Robustness - NVM must continue to perform despite some violations of the assumptions in the specification.

(3) Consistency - there must be traceability of the NVM code to the requirements.

(4) Completeness - the final NVM Tool must be complete.

(5) Self-containedness - the NVM software must perform all of its own error checking.

4.6.3 Efficiency Requirements

NVM should be an efficient tool.

4.6.4 Integrity Requirements

For reasons of security, it shall be necessary for users of the SWG APSE to have different privileges and access rights. Many commands which are needed by every SWG APSE user should be restricted to their own environment, e.g., deletion of nodes is necessary but users should only be able to delete nodes that they created.

The NVM Tool shall adhere to the discretionary and mandatory access control rules imposed by the CAIS implementation in order to control unauthorized access to the data in the SWG APSE database.

Process nodes created by NVM shall be given the default role associated with the user's group node and the adopted roles of any of the groups of which the user is a potential member. This scheme shall ensure that processes created for the user have proper access to the user's node structure.

4.6.5 Usability Requirements

NVM shall provide a user interface that is easy to learn and use, and adaptable to the expertise of the user. It shall not be necessary to know how to tailor the environment in order to use the NVM Tool effectively. The default environment shall provide all capabilities necessary for invoking processes, reporting the status of the SWG APSE database, and manipulating the nodes in the database. However, users of various levels of expertise shall be able to tailor the environment as desired.

On-line help for each command shall be available to the user. The help facility shall provide access to information on the NVM commands. This information shall include at least a summary of the command's function and the command's parameters. Other information may be provided.

4.6.6 Maintainability Requirements

NVM shall be maintainable. The availability of documentation, specifications, and source code shall be sufficient for NVM maintenance.

4.6.7 Testability Requirements

To enable evaluation of the compliance with the NVM Tool requirements, the software test environment shall minimally consist of the following:

(1) A hardware configuration similar to that defined by the SWG APSE requirements.

(2) The host software as defined by the SWG APSE requirements.

(3) The CAIS implementation.

(4) The SWG APSE compiler.

(5) The SWG APSE linker.

(6) The NVM Tool.

The general NVM test requirements shall be as follows:

(1) The NVM Tool size and execution time shall be measured.

(2) The NVM Tool shall be tested using nominal, maximum, and erroneous input values.

(3) The NVM Tool shall be tested for error detection and proper error recovery, including appropriate error messages.

(4) The NVM Tool shall be tested using input data from an interactive user.

(5) The facilities of the NVM Tool shall be tested using sample NVM Command Language scripts.

(6) All NVM commands shall be tested.

4.6.8 Flexibility Requirements

The extensibility of the NVM Tool shall provide a means to enhance NVM. The extension shall consist of modifying the source code of the NVM Tool in order to incorporate the new command(s).

Individual users may also extend or enhance NVM through the use of Command aliases.

4.6.9 Portability Requirements

The NVM interface to the underlying operating system shall be via the defined SWG CAIS implementation.

The main portability consideration for rehosting NVM shall be the terminals being used on the system. The package CAIS_TERMINAL_IO is used by NVM. This package assumes that a screen terminal is present that can correctly interpret ANSI character sequences.

Minimally, the SWG CAIS implementation, the compiler and the linker must be rehosted before NVM may be rehosted. The NVM Tool has a production dependency on the SWG APSE compiler and linker. The features of the Ada language which are compiler dependent may impose portability issues on the NVM Tool.

4.6.10 Reusability Requirements

There are no requirements for reusability. The NVM implementation shall be available in order to be part of the NATO SWG APSE.

4.6.11 Interoperability Requirements

Interfaces to other SWG APSE tools shall be provided through the SWG CAIS.

4.7 Other SWG APSE Tool Support

The development, operation and support of the NVM Tool shall require the hardware and software configurations specified in the NATO SWG APSE Requirements. The minimum toolset needed for the proper operation of NVM shall be the CAIS implementation, the compiler, and the linker.

The development and support for the NVM Tool shall require a proficiency in the Ada programming language.

Documentation on the NVM Tool shall be provided for the support and operation of NVM. All documentation relevant to the operation of NVM shall be available on-line through the NVM help facility.

5. QUALIFICATION REQUIREMENTS

5.1 General Qualification Requirements

Since the NVM Tool is a prototype, there should not be an essential need for software quality assurance. However, software quality assurance should be a part of the implementation of the NVM Tool as much as possible. It should be used to prevent problems from occurring, to remove defects from NVM, to contribute to the usability and maintainability of the software, and to improve the production rate of deliverable code and documentation.

5.2 Special Qualification Requirements

At least the following shall be tested:

(1) The NVM Editing Directives.

(2) All NVM Commands identified in the previous Chapter.

(3) The abbreviated form of a pathname described in this document.

(4) Execution of at least three NVM Commands from the contents of a file node whose pathname is given by **'USER(*user_name*)'DOT(LOGIN)**.

6. PREPARATION FOR DELIVERY

The NVM source code and documentation shall be delivered on a machine readable medium. In addition, at least one hardcopy of the Software Requirements Specification and the User Manual shall be delivered.

7. NOTES

7.1 Acronyms

AD	Attribute Definition
AKD	Attribute Kind Definition
ANSI	American National Standards Institute
APSE	Ada Programming Support System
CAIS	Common APSE Interface Set
DEC	Digital Equipment Corporation
DOD	Department of Defense
MOU	Memorandum of Understanding
NATO	North Atlantic Treaty Organization
NKD	Node Kind Definition
NKI	Node Kind Interpretation
NVM	Node and View Management Tool
RD	Relation Definition
RI	Relation Interpretation
SRS	Software Requirements Specification
SWG	Special Working Group
VAX	Virtual Address Extension
VMS	Virtual Memory System

Appendix A

SAMPLE INTERACTIVE SESSION

The following is an example of dialogue that may occur in an interactive session of the NVM Tool.

Comments below (indicated by "`-- comment`") are not part of the NVM command. They are included for explanation purposes.

```
-- alias some commands so they look like
-- MS-DOS commands

NVM:USER > alias delete deln -- delete node
NVM:USER > alias type lfc -- list file contents
NVM:USER > alias dir lpr -- list primary relationships

-- list the primary relationships below this node
NVM:USER > dir

'USER(USER)'DOT(COLORS)
'USER(USER)'DOT(LOGIN)
'USER(USER)'JOB(A)

-- list the contents of a file node

NVM:USER > type .colors

screen main color white blue primary_rendition
screen info color white green bold
screen info border white green bold
screen cmds color white blue primary_rendition
screen cmds border red blue bold
screen help color yellow blue primary_rendition
screen help border red blue primary_rendition
```

```
-- create a new file node

NVM:USER > crefn file


-- copy this file node to another one

NVM:USER > copy dot(file) 'user(USER).new_file


-- now list the primary relationships
-- below the current one

NVM:USER > dir

'USER(USER)'DOT(COLORS)
'USER(USER)'DOT(FILE)
'USER(USER)'DOT(LOGIN)
'USER(USER)'DOT(NEW_FILE)
'USER(USER)'JOB(A)


-- list the contents of the newly created node

NVM:USER > type dot(new_file)

[contents of file listed here...]


-- delete the newly created file node

NVM:USER > del dot(new_file)
[the above command could also have been "del .new_file"]
```

```
-- end the NVM session

NVM:USER > bye
```

Appendix B

GUIDELINE FOR NVM COMMAND LANGUAGE SYNTAX

The following may be used as a guideline for the syntax of the NVM Command Language. Lexical categories defined here are found in Appendix D of MIL-STD-1838A. The notation used is a form of Backus-Naur Form (BNF):

Words	identify syntactic categories.
[]	identify optional items;
{}	identify items which may be repeated zero or more times;
"|"	separates alternatives; and
"::="	separates the left-hand and right-hand sides of productions.

```
NVM_Command ::= comment_statement | command_statement

comment_statement ::= --{graphic_character}

command_statement ::= NVM_command {parameter_association}

parameter_association ::= [formal_parameter =>] actual_parameter

formal_parameter ::= identifier

actual_parameter ::= identifier
  | integer_literal
  | string_literal
  | list
  | pathname
  | host_filename

pathname ::= path_name
```

```
NVM_command ::= abort
  | alias
  | bye
  | copyn
  | copyt
  | date
  | deln
  | delt
  | expfc
  | help
  | host
  | impfc
  | inv
  | move
  | res
  | spawn
  | susp
  | sysadmin
  | time
  | unalias
  | other_NVM_command

other_NVM_command ::= identifier
```