Natural

Using Natural

Version 6.3.12 for OpenVMS

October 2012

Natural

This document applies to Natural Version 6.3.12 for OpenVMS.

Document ID: NATOV-NNATUXUSING-6312-20121005

Table of Contents

Preface v
1 Invoking and Terminating a Natural Session 1
Invoking Natural 2
Terminating Natural 2
2 Natural Main Menu 5
Description of the Main Menu 6
Programming Modes 7
Overview of Menus 7
3 Using Natural Libraries 11
General Information 12
Steplibs 12
Search Sequence for Object Execution 13
Logging on to a Library 13
4 Commonly Used System Commands 17
System Commands to Create and Modify Source Code 18
System Commands to Store and Delete Objects 18
System Commands to Execute Programs 18
5 Miscellaneous Information 19
Help on Error Messages 20
Natural Editors 20
Asterisk Notation 20
6 Rules and Naming Conventions 23
Object Naming Conventions 24
Library Naming Conventions 25
Naming Conventions for User-Defined Variables 25

Preface

This documentation introduces you to the usage of Natural in an OpenVMS environment.

The instructions and methods described in this documentation relate to the default standards as delivered with the original Natural software. They are not a comprehensive description of all features provided by Natural. For a full description of all options and functions, refer to the corresponding Natural documentation.

The layout of the example screens provided in this documentation and the behavior of Natural described here can differ from your results. For example, a command or message line may appear in a different screen position, or the execution of a Natural command may be protected by security control. The default settings in your environment depend on the system parameters set by your Natural administrator.

This documentation is organized under the following headings:

Invoking and Terminating a Natural Session	How to start and end a Natural session.
Natural Main Menu	A description of the main menu. Information on the programming modes supported by Natural. An overview of the menus available from the main menu.
Using Natural Libraries	How to create a library and how to log on to a library and display its contents.
Commonly Used System Commands	An overview of the system commands used to create and modify source code, store and delete objects, and execute programs.
Miscellaneous Information	How to invoke help on error messages. Brief information on the Natural editors. How to use asterisk notation.
Rules and Naming Conventions	Natural-specific rules and naming conventions for objects, libraries and user-defined variables.

1 Invoking and Terminating a Natural Session

- Invoking Natural 2
- Terminating Natural 2

It is also possible to start and terminate a Natural batch session. See *Natural in Batch Mode* in the *Operations* documentation.

Invoking Natural

The way you invoke Natural depends on how the system has been configured at your site. For most installations, you invoke Natural as described below.

To invoke Natural

- Enter the following command at the DCL prompt:

  ```
  natnn
  ```

  where *nn* is the current version number.

  The main menu appears. See ***Natural Main Menu*** for further information.

Notes:

1. You can set up your own environment by specifying dynamic parameters when starting Natural Studio. See *Dynamic Assignment of Parameter Values* in the *Operations* documentation.
2. If Natural Security is active, access to some libraries and some functions may be restricted. Ask your Natural Security administrator for details.

Terminating Natural

A Natural session can be terminated in different ways.

To terminate Natural

- Select the **Fin** menu and press ENTER.

  Or:

  Select the **Direct** menu and press ENTER. In the resulting **Direct Command** window enter the system command `FIN` and press ENTER.

  Or:

  Execute a Natural program which contains a `TERMINATE` statement.

Note: Termination methods can also be defined with the Configuration Utility. See also *Program Loading and Deletion* in the *Overview of Profile Parameters* section of the *Configuration Utility* documentation.

2 Natural Main Menu

- Description of the Main Menu 6
- Programming Modes 7
- Overview of Menus 7

Description of the Main Menu

When you **invoke** Natural, the main menu is displayed:

```
2008-03-10                          NATURAL                  Library: SYSTEM
 15:12:57                   V 6.3.4 Software AG 2008          Mode   : REPORT
 User: SAG                                                  Work Area : empty
+-------------------------------------------------------------------------------+
¦Library          Direct           Services         OS                Fin      ¦
+-------------------------------------------------------------------------------+









 Select Library
```

The top left-hand corner of the main menu displays the following information:

- The current date and time.
- The current user ID. By default, this is your OpenVMS user ID.

The top right-hand corner of the main menu displays the following information:

- The name of the current library. See also ***Using Natural Libraries***.

  A minus (-) character after the library name indicates that in this library the line numbers are suppressed in the source code which is saved in the file system. The line number suppression state can be changed using the `FTOUCH` utility.
- The current programming mode (reporting mode or structured mode). See also ***Programming Modes*** below.
- The name of the programming object currently in the editor work area. This work area is where Natural places a programming object which is to be edited. The string "empty" indicates that there is no object in the work area.

For information on the menus provided in the main menu, see *Overview of Menus* below.

Programming Modes

Natural supports two programming modes:

- **Structured Mode**
  Structured mode is intended for complex applications with a clear and well-defined program structure. It is recommended to use structured mode exclusively.
- **Reporting Mode**
  Reporting mode is useful only for the creation of ad hoc reports and small programs which do not involve complex data and/or programming constructs.

For more information on the differences between reporting and structured mode, see *Natural Programming Modes* in the *Programming Guide*.

To switch to another mode

- Select the **Direct** menu and press ENTER. In the resulting **Direct Command** window enter the system command `GLOBALS` as described below and press ENTER.

  To switch from reporting mode to structured mode:

  ```
  GLOBALS SM=ON
  ```

  To switch from structured mode to reporting mode:

  ```
  GLOBALS SM=OFF
  ```

Overview of Menus

The following menus are available from the main menu:

- Library
- Direct
- Services
- OS

- Fin

Library

If you choose this menu, you can create a new Natural library or log on to an existing Natural library. For detailed information, see ***Using Natural Libraries***.

Direct

If you choose this menu, the **Direct Command** window is displayed in which you enter the name of the system command or program you wish to be executed (provided the program is in your current library, the steplib or the library `SYSTEM`). See also ***Commonly Used System Commands***.

Natural system commands are used, for example, to:

- find, access, manipulate, check, and compile objects in a Natural library;
- start, end, and display information about a Natural session;
- access Natural utilities.

Notes:

1. You can also enter a system command in response to a `MORE` prompt. In this case, the program that is being executed will be stopped and the system command will be executed.
2. Do not confuse Natural system commands, which are used to perform session functions, with Natural statements, which are the components of Natural programs.

Services

If you choose this menu, a window appears and you can choose one of the following commands:

Command	Purpose
DDM Services	Data Definition Modules (DDMs) can be created/edited, and various other DDM maintenance functions can be performed. For detailed information, see *DDM Services* in the *Editors* documentation.
SYSMAIN	Various library-related functions can be performed. For detailed information, see *SYSMAIN Utility* in the *Tools and Utilities* documentation.

OS

If you choose this menu, a window appears and you can choose the following command:

Command	Purpose
Exit to OS	Creates a subprocess and invokes the DCL prompt. You can then enter an OpenVMS system command or program name.

To return to the current Natural session, enter the command `logout` at the DCL prompt.

Note: If you are not able to create a subprocess, shell access has been disallowed in the Configuration Utility (profile parameter `SHELL`); in this case, contact your Natural administrator.

Fin

If you choose this menu, you are asked whether you want to exit from Natural.

3 Using Natural Libraries

- General Information 12
- Steplibs 12
- Search Sequence for Object Execution 13
- Logging on to a Library 13

For more information on libraries, see *SYSMAIN Utility* in the *Tools and Utilities* documentation.

General Information

Libraries are used to store Natural objects such as programs, subprograms, subroutines, helproutines or text.

Some libraries are delivered with Natural, such as libraries beginning with the prefix "SYS". The library `SYSTEM`, for example, contains system-related programs, error messages and DDMs (data definition modules).

When you first logon to Natural, your default library will in most cases be `SYSTEM`. Since this library generally contains system-related objects, you will probably want to use a different library to store your programs. You can add your own libraries to the Natural environment or use libraries set up for this purpose.

Note: You can alter your default library from `SYSTEM` to any other library by specifying the Natural profile parameter `INIT-LIB` in the Configuration Utility.

Steplibs

A steplib is a Natural user library or system library that is concatenated with the current user or system library. This avoids redundant storage of identical objects and helps organize applications. In addition, if Natural Security is installed, a steplib can be used to restrict access to particular objects.

Natural searches in a steplib when an object is not found in the current library (see ***Search Sequence for Object Execution*** for further information). The standard steplibs are the libraries `SYSTEM` in the `FUSER` and the `FNAT` system files.

If Natural Security is active, you can define additional steplibs in the security profile of each library. The entries in a library security profile override any definitions made outside Natural Security.

If Natural Security is not active, you can specify additional steplibs in the Configuration Utility, with the profile parameter `STEPLIB`. In addition, you can define further steplibs by using one of the application programming interfaces (for example, `USR1025N` or `USR3025N`) that are supplied for this purpose in the Natural system library `SYSEXT`.

The additional steplibs are searched for an object before the standard steplibs `SYSTEM` (`FUSER` and `FNAT`).

The currently active steplibs are shown when you enter the system command `TECH`.

Search Sequence for Object Execution

This section describes the sequence in which Natural libraries and system files are searched for a requested object that is to be executed from either a user library or a system library.

Note: If the profile parameter BPSFI is set to "ON" (the default setting is "OFF"), objects are searched for in the buffer pool first.

The search sequence for a user-written object to be executed from a user library is as follows:

1. The current library in the FUSER system file as defined by the system variable *LIBRARY-ID.
2. The steplibs (in sequence) as specified in the Natural Security profile for the current library or in the steplib table.
3. The default steplib as defined by the system variable *STEPLIB.
4. The library SYSTEM in the FUSER system file.
5. The library SYSTEM in the FNAT system file.

The search sequence for a Natural object to be executed from a system library is as follows:

1. The current "SYS" library in the FNAT system file as defined by the system variable *LIBRARY-ID.
2. The steplibs (in sequence) as specified in the Natural Security profile for the current library or in the steplib table.
3. The library SYSLIBS in the FNAT system file, which contains objects shared by system commands and utilities.
4. The library SYSTEM in the FNAT system file.
5. The library SYSTEM in the FUSER system file.

   Since the FUSER system file is searched last, you must provide an object that is used in both the FUSER and the FNAT system files (for example, a user-exit routine for a Natural utility) only in one location, namely in FUSER.

Logging on to a Library

To log on to a library, select the **Library** menu and press ENTER. A window appears showing the command **<LOGON>** followed by a list of all existing libraries. For example:

```
+----------+
¦ <LOGON>  ¦
¦ DEMO1    ¦
¦ DEMO2    ¦
¦ DEMO3    ¦
¦ SAMPLES  ¦
¦ SYSEXPG  ¦
¦ SYSEXSYN ¦
¦ SYSTEM   ¦
+----------+
```

To create a new library, choose the **<LOGON>** command. A window is then displayed in which you enter the name of the library to be created. See also *Library Naming Conventions*.

To use an existing library, you can either choose the library from the list, or choose the **<LOGON>** command and then enter the name of the desired library. The contents of the library are then displayed. For example:

```
+------------------------------ List * * -------------------------------+
¦ Cmd Name     Type         SM S/C User ID  SRC Date       GP Date        ¦
¦ --- -------- ------------ -- --- -------- -------------- -------------- ¦
¦ <DIRECT COMMAND>                                                       ¦
¦     ARRAYD   Program      S  S   SAGPC    07:45 02-02-07               ¦
¦     ARRAYE   Program      S  S/C SAGPC    07:45 02-02-07 09:25 02-02-07 ¦
¦     BREAK1   Program      S  S/C SAGPC    07:45 02-02-07 09:25 02-02-07 ¦
```

Field	Explanation	
Cmd	In this field, you can enter a function code. Enter "?" or press F2 to display the list of the available function codes for the selected object. The function codes are:	
	C	Check the object's source code for syntax errors.
	D	Read the object's source code into work area.
	E	Edit the object's source.
	L	List the object's source code.
	I	Display the directory information about the object.
	H	Print hardcopy of the object's source.
	R	Run (that is, compile and execute) the object's source.
	X	Execute the object.
	S	Stow the object in source and object form.
	U	Scratch (that is, delete) the object's source and object form.
	.	End.
Name	Object name.	
Type	Object type.	
SM	Programming mode:	

Field	Explanation
	S Structured mode. R Reporting mode.
S/C	S The object exists in source form. C The object exists in object (cataloged) form.
User ID	The ID of the user who saved/cataloged the object.
SRC Date	Source program date. The time and date when the object was last saved.
GP Date	Generated program date. The time and date when the object was last cataloged.

Note: It is also possible to enter the system command `LOGON library-name` in the **Direct Command** window.

4 Commonly Used System Commands

- System Commands to Create and Modify Source Code 18
- System Commands to Store and Delete Objects 18
- System Commands to Execute Programs 18

For a detailed description of each system command, see the *System Commands* documentation.

System Commands to Create and Modify Source Code

System Command	Purpose
EDIT	Edit the source form of an object.
CLEAR	Clear the contents of the work area of the current editor. The source code currently in the work area is not saved.
CHECK	Check the source code of an object for syntax errors. Syntax checking is also performed as part of the RUN and STOW commands.

System Commands to Store and Delete Objects

System Command	Purpose
SAVE	Save the source form of the Natural object currently in the work area of the editor and store it. Syntax is not checked. A saved program can be run, but not executed (see the corresponding system commands below).
STOW	Save the source form of an object, compile the object and store the resulting object module as well as the source. The object is syntax checked during the compilation process.
SCRATCH	Delete the source and object form of an object. A list of all objects stored in the current library will be displayed; on the list you may then mark the object(s) to be deleted.
PURGE	Delete the source form of an object. A list of all objects stored in the current library will be displayed; on the list you may then mark the object(s) to be deleted.
UNCATALOG	Delete the object form of an object. A list of all objects stored in the current library will be displayed; on the list you may then mark the object(s) to be deleted.

System Commands to Execute Programs

System Command	Purpose
RUN	Compile and execute a source program, but not a program stored in object form.
EXECUTE	Execute a program that has been compiled and stored in object form.

5 Miscellaneous Information

- Help on Error Messages 20
- Natural Editors 20
- Asterisk Notation 20

Help on Error Messages

To get more information on an error message displayed by Natural, issue the following system command:

```
HELP nnnn
```

where *nnnn* is the number of the error message.

An extended message text will be displayed, which provides more detailed information about the error. See also the description of the system command HELP.

Natural Editors

The editors provided with Natural are described in the *Editors* documentation.

The editors are invoked with the system command EDIT. The editor invoked depends on the type of object you specify. If you specify an object by name, the appropriate editor is automatically invoked.

When you switch from one editor to another, all changes are lost unless previously saved. The editing area is overwritten by the new object to be edited.

See also *First Steps with Natural*. This tutorial provides a very simple and brief introduction to programming with Natural and to using the Natural editors.

Asterisk Notation

Many Natural functions display lists of objects. Usually these lists contain all objects available (for example, all objects of a given type, all objects in a given library, etc.). If you do not wish all objects to be listed, but only a certain range of objects, you may specify that range by using asterisk notation.

By specifying a parameter value followed by an asterisk (*) you can get a list of only those objects whose names (or IDs or whatever the parameter is) begin with that value. This option to enter a value followed by an asterisk is referred to as asterisk notation.

Example 1

If you enter the system command SCRATCH without any parameters:

```
SCRATCH
```

you will get a list of all objects in the current library. You can then mark the objects which are to be deleted.

Example 2

If you enter the system command `SCRATCH` as follows:

```
SCRATCH BOC*
```

you will get a list of only those objects in the current library whose names begin with "BOC". You can then mark the objects which are to be deleted.

6 Rules and Naming Conventions

- Object Naming Conventions 24
- Library Naming Conventions 25
- Naming Conventions for User-Defined Variables 25

This section describes Natural-specific rules and naming conventions.

Object Naming Conventions

This section describes the naming conventions that apply when saving and/or cataloging a Natural object in a Natural system file.

The name of a Natural object can be 1 to 8 characters (listed in the following table) where the *first* character must be one of the following:

- an upper-case alphabetical character
- a number sign (#)
- a plus sign (+)

If the first character is a number sign (#) or a plus sign (+), the name must consist of at least one additional character.

Exception:

The name of a Natural DDM can be 1 to 32 characters (listed in the following table) where the *first* character must be an upper-case alphabetical character.

The name of a Natural object can consist of the following characters:

Character	ISO Character Name	Remark
A - Z	Latin capital letter A - Z	Upper-case alphabetical character
0 - 9	Digit zero - digit nine	Numeric character
-	Hyphen-minus	Hyphen
_	Low line	Underscore
/	Solidus	Slash
@	Commercial at	
$	Dollar sign	
&	Ampersand	Only allowed in language codes See also *Defining the Language of a Natural Object* in the *Programming Guide*.
#	Number sign	Hash sign
+	Plus sign	Only allowed as the first character

Library Naming Conventions

This section describes the naming conventions that apply to a Natural library.

The name (ID) of a library can be 1 to 8 characters and must *not* start with "SYS". The prefix "SYS" is reserved for Natural system libraries.

A library name must start with an upper-case alphabetical character. Any other alphabetical character must also be upper case. A library name must *not* contain any blank characters.

A library name can consist of the following characters:

Character	ISO Character Name	Remark
A - Z	Latin capital letter A - Z	Upper-case alphabetical character
0 - 9	Digit zero - digit nine	Numeric character
-	Hyphen-minus	Hyphen
_	Low line	Underscore

Naming Conventions for User-Defined Variables

This section describes the naming conventions that apply to a user-defined variable:

- Length of Variable Names
- Limitations of Variable Names
- Characters Allowed in Variable Names
- First Character of Variable Names
- Case of Characters in Variable Names

For further information on user-defined variables, refer to the section *User-Defined Variables* in the *Programming Guide*.

Length of Variable Names

The name of a user-defined variable can be 1 to 32 characters long.

You can use variable names of over 32 characters (for example, in complex applications where longer meaningful variable names enhance the readability of programs); however, only the first 32 characters are significant and must therefore be unique, the remaining characters will be ignored by Natural.

Limitations of Variable Names

The name of a user-defined variable must *not* be a Natural reserved keyword.

Within one Natural program, you must *not* use the same name for a user-defined variable and a database field, because this might lead to referencing errors (see *Qualifying Data Structures* in the *Programming Guide*).

Characters Allowed in Variable Names

The name of a user-defined variable can consist of the following characters:

Character	ISO Character Name	Remark
A - Z	Latin capital and/or small letter A - Z	Upper-case and/or lower-case alphabetical character Lower-case *not* allowed as the first character
0 - 9	Digit zero - digit nine	Numeric character
-	Hyphen-minus	Hyphen
_	Low line	Underscore
/	Solidus	Slash
@	Commercial at	
$	Dollar sign	
&	Ampersand	
#	Number sign	Hash sign
+	Plus sign	Only allowed as the first character

First Character of Variable Names

The first character of the name must be one of the following:

Character	ISO Character Name	Remark
A - Z	Latin capital letter A - Z	Upper-case alphabetical character
&	Ampersand	
#	Number sign	Hash sign
+	Plus sign	

If the first character is a number sign (#), a plus sign (+) or an ampersand (&), the name must consist of at least one additional character.

Variables in a global data area (GDA) with a plus sign (+) as the first character must be defined at Level 1; see also *Global Data Area* in the *Programming Guide*. Other levels are only used in a redefinition.

A plus sign (+) as the first character of a name is only allowed for application-independent variables (AIVs) and variables in a global data area (GDA).

Names of AIVs must begin with a plus sign (+); see also *Defining Application-Independent Variables* in the description of the `DEFINE DATA` statement in the *Statements* documentation.

An ampersand (&) as the first character of a name is used in conjunction with dynamic source program modification (see the `RUN` statement in the *Statements* documentation), and as a dynamically replaceable character when defining processing rules; see the relevant description in the *Map Editor* documentation.

Case of Characters in Variable Names

With Natural for Windows, UNIX and OpenVMS, lower-case characters entered as part of a variable name are internally converted to upper case.

Caution: If you use lower-case characters as part of the variable name, variable names must be unique regardless of their case.