

# Natural for OpenVMS

## System Commands

Version 6.3.8 for OpenVMS

February 2010

This document applies to Natural Version 6.3.8 for OpenVMS.

Specifications contained herein are subject to change and these changes will be reported in subsequent release notes or new editions.

Copyright © 1984-2010 Software AG, Darmstadt, Germany and/or Software AG USA, Inc., Reston, VA, United States of America, and/or their licensors.

The name Software AG, webMethods and all Software AG product names are either trademarks or registered trademarks of Software AG and/or Software AG USA, Inc. and/or their licensors. Other company and product names mentioned herein may be trademarks of their respective owners.

Use of this software is subject to adherence to Software AG's licensing conditions and terms. These terms are part of the product documentation, located at <http://documentation.softwareag.com/legal/> and/or in the root installation directory of the licensed product(s).

This software may include portions of third-party products. For third-party copyright notices and license terms, please refer to "License Texts, Copyright Notices and Disclaimers of Third-Party Products". This document is part of the product documentation, located at <http://documentation.softwareag.com/legal/> and/or in the root installation directory of the licensed product(s).

## Table of Contents

1 System Commands .....	1
2 Issuing System Commands .....	3
Command Input .....	4
Command Line .....	4
NEXT Prompt .....	4
MORE Prompt .....	4
3 System Command Syntax .....	5
Syntax Elements .....	6
Example of Command Syntax .....	7
4 System Commands Grouped by Function .....	9
Navigating within Natural .....	10
Environment Settings .....	10
Editing and Storing Programming Objects .....	11
Executing Programs .....	11
Maintenance Utilities .....	12
Transfer of Programming Objects .....	12
Monitoring and Debugging .....	12
Miscellaneous .....	12
5 CATAL .....	15
CATAL in Interactive Mode .....	16
CATAL in Batch Mode .....	17
6 CATALOG .....	19
7 CHECK .....	21
8 CLEAR .....	23
9 COMPOPT .....	25
Syntax Explanation .....	26
Compiler Options .....	26
Specifying Compiler Parameters .....	26
10 DEBUG .....	29
11 EDIT .....	31
Syntax 1 .....	32
Syntax 2 .....	34
Syntax 3 .....	34
12 EXECUTE .....	35
Syntax Explanation .....	36
Examples of EXECUTE Command .....	37
13 FIN .....	39
14 GLOBALS .....	41
Syntax Explanation .....	42
List of Parameters .....	42
Interaction with SET GLOBALS and Other Statements .....	43
15 HELP .....	45
16 INPL .....	47

17 KEY .....	49
Assigning Commands .....	50
Activating/Deactivating All Keys - KEY ON/OFF .....	51
Activating/Deactivating Individual Keys - KEY key=ON/OFF .....	51
18 LAST .....	53
19 LASTMSG .....	55
20 LIST .....	57
Syntax Overview .....	58
Displaying an Individual Source .....	59
Displaying a List of Objects .....	60
Displaying Directory Information .....	61
Displaying Views .....	62
Displaying File Information of Resource Objects .....	62
Displaying File Information of Error Message Containers .....	62
21 LIST COUNT .....	63
22 LIST XREF .....	65
23 LOGOFF .....	67
24 LOGON .....	69
25 MAIL .....	71
26 PROFILE .....	73
27 PURGE .....	75
28 READ .....	77
29 RENAME .....	79
30 RENUMBER .....	81
31 RETURN .....	83
32 RPCERR .....	85
33 RUN .....	87
34 SAVE .....	89
35 SCAN .....	91
Menu Options .....	92
Subcommands .....	93
SCAN under Natural Security .....	94
36 SCRATCH .....	95
37 SETUP .....	97
Syntax Explanation .....	98
SETUP/RETURN Example .....	99
38 STOW .....	101
39 STRUCT .....	103
Indentation of Source Code Lines .....	104
40 SYSDDM .....	107
41 SYSERR .....	109
42 SYSEXT .....	111
43 SYSEXV .....	113
44 SYSFILE .....	115
45 SYSMAIN .....	117

---

46	SYSNCP .....	119
47	SYSOBJH .....	121
48	SYSPROD .....	123
49	SYSPROF .....	125
50	SYSRPC .....	127
51	SYSWIZDB .....	129
52	SYSWIZDW .....	131
53	TECH .....	133
54	UNCATALOG .....	135
55	UPDATE .....	137
56	XREF .....	139

---





# 1 System Commands

---

This documentation describes the Natural system commands.

Natural system commands perform functions you need to create, maintain or execute Natural programming objects. In addition, Natural system commands are used to monitor and administer your Natural environment.

This documentation is organized under the following headings:

 <b>Issuing System Commands</b>	Describes the general rules that apply when you enter a Natural system command.
 <b>System Command Syntax</b>	Explains the symbols that are used within the syntax descriptions of Natural system commands.
 <b>System Commands Grouped by Function</b>	Provides an overview of the Natural system commands grouped according to their functions.
 System Commands in Alphabetical Order	Descriptions of the system commands in alphabetical order.





# 2 Issuing System Commands

---

- Command Input ..... 4
- Command Line ..... 4
- NEXT Prompt ..... 4
- MORE Prompt ..... 4

## Command Input

---

You can issue a system command by entering it in one of the following ways:

- In the **command line**;
- At the Natural **NEXT** or **MORE** prompt.

The following rules apply:

- Command input is not case-sensitive.
- Commands are context-sensitive.
- Some Natural commands affect objects other than the currently active object.

For an explanation of the symbols that are used within the syntax descriptions, see *System Command Syntax*.

## Command Line

---

If you choose the **Direct** menu from the main menu, the **Direct Command** window is displayed in which you enter the system command.

## NEXT Prompt

---

The **NEXT** prompt appears in a Natural application or program when no more output is pending.

## MORE Prompt

---

The **MORE** prompt is displayed at the bottom of an output screen to signal that more output is pending. When a system command is entered in response to a **MORE** prompt, program execution is interrupted and the system command is executed.

# 3 System Command Syntax

---

- Syntax Elements ..... 6
- Example of Command Syntax ..... 7

## Syntax Elements

The following symbols are used within the syntax descriptions of system commands:

Element	Explanation
ABCDEF	Upper-case letters indicate that the term is either a Natural keyword or a Natural reserved word that must be entered exactly as specified.
<u>ABCDEF</u>	If an optional term in upper-case letters is completely underlined (not a hyperlink!), this indicates that the term is the default value. If you omit the term, the underlined value applies.
<u>ABCDEF</u>	If a term in upper-case letters is partially underlined (not a hyperlink!), this indicates that the underlined portion is an acceptable abbreviation of the term.
<i>abcdef</i>	Letters in italics are used to represent variable information. You must supply a valid value when specifying this term.
[ ]	Elements contained within square brackets are optional.  If the square brackets contain several lines stacked one above the other, each line is an optional alternative. You may choose at most one of the alternatives.
{ }	If the braces contain several lines stacked one above the other, each line is an alternative. You must choose exactly one of the alternatives.
	The vertical bar separates alternatives.
...	A term preceding an ellipsis may optionally be repeated. A number after the ellipsis indicates how many times the term may be repeated.  If the term preceding the ellipsis is an expression enclosed in square brackets or braces, the ellipsis applies to entire bracketed expression.
,...	A term preceding a comma-ellipsis may optionally be repeated; if it is repeated, the repetitions must be separated by commas. A number after the comma-ellipsis indicates how many times the term may be repeated.  If the term preceding the comma-ellipsis is an expression enclosed in square brackets or braces, the comma-ellipsis applies to entire bracketed expression.
:...	A term preceding a colon-ellipsis may optionally be repeated; if it is repeated, the repetitions must be separated by colons. A number after the colon-ellipsis indicates how many times the term may be repeated.  If the term preceding the colon-ellipsis is an expression enclosed in square brackets or braces, the colon-ellipsis applies to entire bracketed expression.
Other symbols (except [ ] { }   ... ,... :...)	All other symbols except those defined in this table must be entered exactly as specified.  <b>Exception:</b>  The SQL scalar concatenation operator is represented by two vertical bars that must be entered literally as they appear in the syntax definition.

## Example of Command Syntax

---

`CATALOG [object-name [library-id]]`

- CATALOG is a Natural keyword which you must enter as specified. The underlining indicates that you may also enter it in abbreviated form as CAT.
- *object-name* and *library-id* are user-supplied operands for which you specify the name of the program you wish to deal with and the ID of the library in which that program is contained.
- The square brackets indicate that *object-name* and *library-id* are optional elements which you can, but need not, specify. The grouping of the brackets indicate that you can specify CATALOG alone, or CATALOG followed either by a program name only or by a program name and a library ID; however, you cannot specify a library ID if you do not also specify a program name.



# 4 System Commands Grouped by Function

---

- Navigating within Natural ..... 10
- Environment Settings ..... 10
- Editing and Storing Programming Objects ..... 11
- Executing Programs ..... 11
- Maintenance Utilities ..... 12
- Transfer of Programming Objects ..... 12
- Monitoring and Debugging ..... 12
- Miscellaneous ..... 12

This chapter provides an overview of the Natural system commands grouped according to their functions.

## Navigating within Natural

---

Command	Function
FIN	Terminates a Natural session.
LOGOFF	Causes the library ID to be set to SYSTEM and the Adabas password to be set to blanks. The contents of the source program work area are not affected by this command.
LOGON	Establishes a library ID for the user. In the specified library, all source or object programs saved during the session will be stored (unless you explicitly specify another library ID in a SAVE, CATALOG or STOW command).
RETURN	Returns to a return point set by a SETUP command.
SETUP	Establishes a return point to which control can be returned using a RETURN command. This allows you to easily transfer from one application to another during a Natural session.

## Environment Settings

---

Command	Function
COMPOPT	Sets various compilation options that affect the way in which Natural programming objects are compiled.
GLOBALS	Changes the settings of various Natural session parameters.
KEY	Assigns functions to keys to be used in your Natural session.
PROFILE	Only available if Natural Security has been installed.  Displays the security profile currently in effect. This profile informs you of the conditions of use in effect for you in your current Natural environment.
SYSPROD	Displays a list of the products installed at your site, and some information on these products.
SYSPROF	Displays the current definitions of the Natural system files.



## Editing and Storing Programming Objects

Command	Function
CATALL	Catalogs <i>all</i> objects or selected objects in the current library.
CATALOG	Compiles the Natural programming object currently in the source work area of an editor, and if the syntax has been found to be correct, stores the resulting object module in the Natural system file.
CHECK	Checks that the source code of a programming object does not contain any syntax errors. The checking process varies according to the type of object being checked.  Syntax checking is also performed as part of the system commands <a href="#">RUN</a> , <a href="#">CATALL</a> , <a href="#">CATALOG</a> and <a href="#">STOW</a> .
CLEAR	Clears the contents of the work area of the editor.
EDIT	Edits the source form of a programming object.
LIST	Lists one or more objects which are contained in the current library.
READ	Transfers an object in source form from the Natural system file to the source work area.
RENUMBER	Renumbers the source code which is currently held in the source work area.
SAVE	Stores the <i>source form</i> of the programming object currently in the work area of the editor in the Natural system file.
SCAN	Searches for a string of characters within an object, with an option to replace the string with another string.
STOW	Compiles and stores a Natural programming object (in both source and object form) in the Natural system file.
STRUCT	Performs structural indentation of a source program, and helps detecting structural inconsistencies.
SYSWIZDW	Invokes the Natural Dialog Wizard, a tool for creating dialogs for specific purposes. The defined dialogs can have several layouts that adapt to desired requirements.

## Executing Programs

Command	Function
EXECUTE	Executes a program that has been compiled and stored in object form. You can EXECUTE a program only if it has been stored in compiled form.
RUN	Compiles and executes the source program currently in the work area of the editor.

## Maintenance Utilities

---

Command	Function
<a href="#">SYSDDM</a>	Creates and maintains Natural data definition modules (DDMs).
<a href="#">SYSERR</a>	Creates and maintains the messages you wish your Natural applications to display to the users.
<a href="#">SYSNCP</a>	Creates and maintains the command processors to be used in your Natural applications.
<a href="#">SYSRPC</a>	Creates and maintains remote procedure calls, that is, provides the settings necessary to execute a subprogram located on a remote server.

## Transfer of Programming Objects

---

Command	Function
<a href="#">SYSMAIN</a>	Transfers objects within the Natural system from one library to another.
<a href="#">SYSOBJH</a>	Processes Natural and non-Natural objects for distribution in Natural environments.

## Monitoring and Debugging

---

Command	Function
<a href="#">RPCERR</a>	Displays the last Natural error number and message if related to Remote Procedure Call (RPC), and the last Broker reason code and associated message.
<a href="#">TECH</a>	Displays technical and other information on your Natural session.

## Miscellaneous

---

Command	Function
<a href="#">HELP</a>	Invokes the Natural help system.
<a href="#">INPL</a>	Invokes the INPL utility. It is <i>only</i> used for the loading of Software AG installation datasets into the system files.
<a href="#">LAST</a>	Displays the system commands that were last executed, and allows you to execute them again.
<a href="#">LASTMSG</a>	Displays additional information on the error situation which occurred last.

Command	Function
MAIL	<p>Only available if Natural Security has been installed.</p> <p>Invokes a mailbox to modify its contents and/or expiration date. A mailbox is used as a notice board to broadcast messages to Natural users.</p>
SYSEXT	Invokes the library SYSEXT, which contains various Natural user application interfaces.
SYSEXV	Invokes the SYSEXV application with examples of the new features of the current Natural versions.
SYSFILE	Invokes the function Natural Print/Work Files of the SYSFILE utility. This utility provides information on the work files and print files available.
SYSWIZDB	Invokes the Natural Data Browser, a development tool wizard within Natural Studio. It enables you to display and print or store file structures.
UPDATE	Prevents database updating being carried out by a program.
XREF	<p>Only available if Predict has been installed.</p> <p>Controls the usage of the Predict function “active cross-references”. This function automatically creates documentation in Predict about the objects which a program/data area references.</p>



# 5 CATALL

---

▪ CATALL in Interactive Mode .....	16
▪ CATALL in Batch Mode .....	17

This command is used to catalog, check, save or stow all objects or selected objects in the current library.

## CATALL in Interactive Mode

### CATALL


When you issue this command, the **Catalog Objects in Library** window appears. In this window, you specify which types of objects are to be processed. Objects are processed in the order in which the object types are listed in the dialog box. Additionally, you can choose which action is to be performed and which objects are to be processed.

You can make the following specifications in the window:

<b>Starting from</b>	<p>Enter an asterisk (*) if you want to process all objects of the selected types in the current library.</p> <p>If you want to restrict the number of objects to a certain range, you can use asterisk notation for the name.</p>
<b>Apply action only to existing modules</b>	<p>If you mark this option, only those objects for which object modules already exist in the current library will be cataloged again; objects that exist only in source form will not be processed.</p>
<b>Apply action to all sources</b>	<p>If you mark this option, <i>all</i> selected objects will be processed.</p>
<b>Action</b>	<p>After pressing PF2, you can select one of the following actions to be applied to the selected objects:</p> <ul style="list-style-type: none"> <li>■ <b>Catalog</b></li> <li>■ <b>Check</b></li> <li>■ <b>Save</b></li> <li>■ <b>Stow</b></li> </ul> <p>These actions correspond to the system commands of the same names.</p> <p><b>Note:</b> Under Natural Security, certain actions may be disallowed.</p>
<b>Renumber source lines</b>	<p>By default, the source-code lines of sources that were saved or stowed are also renumbered.</p> <p>If you wish no automatic renumbering of lines, overwrite the X in this field with a blank.</p>
<b>Object types</b>	<p>By default, CATALL applies to objects of all types in the current library (all object types are marked with an X). If you wish objects of a certain type <i>not</i> to be affected by CATALL, overwrite the corresponding X with a blank.</p>

<b>Generate new map source</b>	Maps created with previous Natural versions are not necessarily compatible with Natural Version 3.1 and above. Mark this option to ensure that maps are converted during the catalog operation. This option converts and stores maps in source <i>and</i> object form.
--------------------------------	--

During CATALL processing, a statistics window appears and the objects being cataloged are listed.

 **Caution:** If you press any key while CATALL processing is in progress, CATALL will be stopped.

Upon successful completion of processing, an information message is displayed.

If an object was not cataloged successfully, a window showing the object name, error number and error line is displayed.

If the CATALL command is called from within a Natural program with stacked parameters, Natural assumes batch mode and does not display the error dialog in case of compilation errors. Instead of the dialog a file will be created in Natural's temporary directory (TMP\_PATH) with name *library.CTL*. This file is empty if no errors are found, otherwise it contains the failed objects with line number and error code.

## CATALL in Batch Mode

```
CATALL object-name [ RECAT ] [TYPES types] [ CATALOG ] [options ...]
                   [ ALL ]
```

For the various specifications you can make in the **Catalog Objects in Library** window, there are also corresponding options which you can specify directly with the system command CATALL:

<i>object-name</i>	The name of the object to be cataloged.  Enter an asterisk (*) if you want to catalog all objects of the specified types in the current library.  If you want to restrict the number of objects to a certain range, you can use asterisk notation for the name.
<b>RECAT / ALL</b>	Corresponds to the options <b>Apply action only to existing modules</b> , or <b>Apply action to all sources</b> of the <b>Catalog Objects in Library</b> screen. RECAT is the default.
<b>TYPES</b> <i>types</i>	Corresponds to the object types marked in the <b>Catalog Objects in Library</b> screen. Possible <i>types</i> are:  G Global data areas

	<p>A Parameter data areas                  L Local data areas                  D DDM                  S Subroutine                  N Subprogram                  H Helproutine                  M Map                  P Program                  3 Dialog                  4 Class                  7 Function                  8 Adapter                  * All types (this is the default)</p> <p>The <i>types</i> have to be specified as <i>one</i> character string, for example, LAG for local, parameter and global data areas. By default, CATALL applies to objects of all types in the current library.</p>	
<b>CATALOG / CHECK / SAVE / STOW</b>	Corresponds to the actions of the same names on the <b>Catalog Objects in Library</b> screen. CATALOG is the default.	
<i>options</i>	NOREN	No automatic renumbering of source-code lines of source objects.



**Note:** The individual command components must be separated from one another either by a blank or by the input delimiter character (as defined with the session parameter ID).




# 6 CATALOG

---


```
CATALOG [object-name [library-id]]
```

Related commands: [SAVE](#) | [STOW](#) | [UNCATALOG](#).

This command is used to compile the Natural programming object currently in the source work area of an editor and (if the syntax has been found to be correct) store the resulting object module in the Natural system file.

 **Important:** The CATALOG command cannot be used if the profile parameter RECAT has been set to ON; in this case, use the [STOW](#) command to compile and store the object.

<b>CATALOG</b>	If you do not specify an <i>object-name</i> , the object is cataloged in the current library under the name of the object last read into the source work area (for example, with <code>EDIT</code> or <code>READ</code> ). An existing object code will be replaced.
<b>CATALOG</b> <i>object-name</i>	A new object is created. As <i>object-name</i> , you specify the name under which the new object is to be cataloged. It is stored in the current library. If the object exists, the command is rejected.
<b>CATALOG</b> <i>object-name</i> <i>library-id</i>	If you want the new object to be cataloged into another library, you must specify the <i>library-id</i> of that library. If the object exists, the command is rejected.

 **Note:** If an FDIC system file is specified in the parameter file which is not valid, Natural will display an appropriate error message when the CATALOG command is issued.



# 7 CHECK

---

CHECK

This command is used to check if the syntax of the source code currently in the editor work area contains any errors.

If a syntax error is detected, syntax checking is suspended and the line containing the error is displayed. You can then either correct the line (whereupon verification continues) or press ENTER without modifying the line displayed. This stops the verification procedure and opens the editor.



**Note:** Syntax checking is also performed as part of the [RUN](#), [STOW](#), [CATALOG](#) and [CATALL](#) commands.



# 8 CLEAR

---

CLEAR

This command is used to clear the source work area of the editor. It can be used if a new program is to be created and there is another object in the source work area.



# 9 COMPOPT

---

▪ Syntax Explanation .....	26
▪ Compiler Options .....	26
▪ Specifying Compiler Parameters .....	26

```
COMPOPT [option=value ...]
```

This system command is used to set various compilation options. The options are evaluated when a Natural programming object is compiled.

If you enter the COMPOPT command without any options, a screen is displayed where you can enable or disable the options described below.

The default settings of the individual options are set with the corresponding profile parameters in the Natural parameter file.

## Syntax Explanation

---

<b>COMPOPT</b>	If you issue the COMPOPT system command without options, the <b>Compilations Options</b> screen appears. The keywords available there are described below.
<b>COMPOPT</b> <i>option=value</i>	Instead of changing an option on the screen, you can also specify it directly with the COMPOPT command.  <b>Example:</b> <pre>COMPOPT DBSHORT=ON</pre>

## Compiler Options

---

The following compiler options are available. For details on the purpose of these options and the possible settings, see the description of the corresponding Natural profile parameter:

KCHECK | PCHECK | DBSHORT | PSIGNF | TQMARK | THSEP | GFID | MASKCME

## Specifying Compiler Parameters

---

You can specify compiler parameters on different levels:

### 1. As Default Settings

The default settings of the individual compiler parameters are specified using the **Compiler Options** category of the Configuration Utility and are stored in the Natural parameter file NATPARM.



## 2. At Session Start

At session start, you can override the compiler option settings by specifying the corresponding profile parameters.

## 3. During an Active Natural Session

During an active Natural session, there are two ways to change the compiler parameter values with the COMPOPT system command: either directly using command assignment (`COMPOPT option=value`) or by issuing the COMPOPT command without options which displays the **Compilation Options** screen. The settings assigned to a compiler option are in effect until you issue the next LOGON command to another library. At LOGON to a different library, the default settings (see item 1 above) will be resumed. Example:

```
OPTIONS KCHECK=ON
DEFINE DATA LOCAL
1 #A (A25) INIT <'Hello World'>
END-DEFINE
WRITE #A
END
```

## 4. In a Natural Programming Object

In a Natural programming object (for example: program, subprogram), you can set compiler parameters with the OPTIONS statement. Example:

```
OPTIONS KCHECK=ON
WRITE 'Hello World'
END
```

The compiler options defined in an OPTIONS statement will only affect the compilation of this programming object, but do not update settings set with the command COMPOPT.



# 10

## DEBUG

---

`DEBUG object-name`

This command is used to invoke the Natural debugger. With the command, you specify the name of the object to be debugged.

You can use the debugger only in a remote development environment (SPoD), using Natural Studio.



**Note:** This command is not executable in batch mode.



# 11 EDIT

---

▪ Syntax 1 .....	32
▪ Syntax 2 .....	34
▪ Syntax 3 .....	34

This command is used to invoke a Natural editor for the purpose of editing the source form of a Natural programming object.

Three different forms of command syntax exist. These are documented in the following sections.

Related command: [READ](#).

See also *Object Naming Conventions* in the *Using Natural* documentation.

## Syntax 1

---

```
EDIT [object-type] [object-name] [library-id]
```

*object-type*

The following object types can be edited:

```
{ CLASS }
  4
COPYCODE
GLOBAL
HELPROUTINE
LOCAL
MAP
PARAMETER
PROGRAM
{ SUBPROGRAM }
  N
SUBROUTINE
TEXT
VIEW
7 (for Function)
```

Which editor is invoked depends on the type of object to be edited:

- Local data areas, global data areas or parameter data areas are edited with the data area editor.
- Maps are edited with the map editor.
- Classes are edited with the program editor.

- `EDIT VIEW` only works in the current library and when an *object-name* is specified. If the object to be viewed is a DDM, the DDM Services are invoked.
- All other types of objects - program, subprogram, subroutine, 7 (for function), helproutine, copycode, text, description - are edited with the program editor.



**Note:** The text object “description” is available on mainframes only. A description is a program description as stored and maintained in the Predict Data Dictionary; an object of this type can only be edited if Predict is installed.

The object types are described in the *Programming Guide*. The editors are described in the *Editors* documentation.



**Note:** If you are in the program editor and enter `EDIT object-name` (whereas the Natural object is of type copycode, class, 7 (for function), helproutine, program, subprogram, subroutine, text), a parallel edit session will be opened. See also the description of the program editor command `NEXT`.

If you specify the name of the object you wish to edit, you need not specify its object type.

*object-name*

With the `EDIT` command, you specify the name of the object you wish to edit. The maximum length of the object name is 8 characters.



**Note:** For DDMs, the maximum length is 32 characters.

Natural will then load the object into the edit work area of the appropriate editor and set the object name for a subsequent `SAVE`, `CATALOG`, `STOW` command.

If you do not specify an *object-name* and there is no object in the source work area, the empty program editor screen will be invoked where you can create a program. If the source work area is not empty, the object will be loaded in the appropriate editor.

*library-id*

If the object you wish to edit is not contained in the library you are currently logged on to, you must specify the *library-id* of the library in which the object to be edited is contained.

If Natural Security is active, a *library-id* must not be specified, which means that you can only edit objects which are in your current library.

## Syntax 2

```
EDIT [ object-type* ] { object-name* }
```

If you do not remember the name of the object you wish to edit, you can use this form of the `EDIT` command to display a list of objects, and then select from the list the desired object.

<code>EDIT *</code>	displays a list of all objects in your current library.
<code>EDIT <i>object-type</i>*</code>	displays a list of all objects of that type in your current library.

To select an object from a certain range of objects, you can use asterisk notation and wildcard notation for the *object-name* in the same manner as described for the system command .

## Syntax 3

```
EDIT FUNCTION subroutine-name
```

The `EDIT FUNCTION` command may be used to edit a subroutine using the subroutine name (not the object name) with maximally 32 characters.



**Note:** Please note that the keyword `FUNCTION` used in this syntax is not identical with the Natural **object type** 7 (for function) listed above. See the description of object type Function in the *Programming Guide*.

Example:

```
DEFINE SUBROUTINE CHECK-PARAMETERS
...
END-SUBROUTINE
END
```

Assuming that the above subroutine has been saved under the object name `CHCKSUB`, you may edit subroutine `CHECK-PARAMETERS` either by issuing the command:

```
EDIT S CHKSUB
```

or by

```
EDIT F CHECK-PARAMETERS
```



# 12 EXECUTE

---

▪ Syntax Explanation .....	36
▪ Examples of EXECUTE Command .....	37

```
{ EXECUTE [REPEAT]    program-name    [library-id] }
{ program-name [parameter ...] }
```

This command is used to execute a Natural object module of type program. The object module must have been cataloged (that is, stored in object form) in the Natural system file or linked to the Natural nucleus. The execution of an object module does not affect the source code currently in the editor work area.

## Syntax Explanation

<b>EXECUTE</b>	The keyword EXECUTE is optional; it is sufficient to specify <i>program-name</i> , i.e. the name of the program to be executed.
<b>REPEAT</b>	If the program being executed produces multiple screen output and you wish the screens to be output one after another without intervening prompts, you specify the keyword REPEAT together with the keyword EXECUTE.
<i>program-name</i>	The name of the program to be executed. If you do not specify a <i>library-id</i> , Natural can only execute the specified program if it is stored either in your current library or in the current steplib library (the default steplib is SYSTEM).
<i>library-id</i>	If the program is stored in another library, specify the <i>library-id</i> of that library. In this case, the program can only be executed if it is actually stored in the specified library.  A <i>library-id</i> that begins with SYS must not be specified (except SYSTEM).
<i>parameter</i>	When you execute a program by specifying the program name without the keyword EXECUTE, you may pass parameters to the program. These parameters will be read by the first INPUT statement in the executed program.  You can specify the parameters as positional parameters or as keyword parameters, with the individual specifications separated from one another by blanks or the input delimiter character (as specified with the session parameter ID).  <b>Note:</b> If one of the parameters passed contains blanks or is a string which contains blanks, the transfer will only be executed if directly after the program name an input delimiter is set.

---

## Examples of EXECUTE Command

---

```
EXECUTE PROG1
```

```
EXECUTE PROG1 ULIB1
```

```
PROG1
```

```
PROG1 VALUE1 VALUE2 VALUE3
```

```
PROG1 VALUE1, VALUE2, VALUE3
```

```
PROG1 PARM1=VALUE1, PARM2=VALUE2, PARM3=VALUE3
```

```
PROG1 PARM3=VALUE3 PARM1=VALUE1 VALUE2
```

```
PROG1,ab cd ef,gh,de fg,ab
```



# 13 FIN

---

`FIN`

This command is used to terminate a Natural session. It applies to online sessions as well as batch mode sessions.

A batch mode session is also terminated when an end-of-file condition is detected in the command input dataset.



# 14 GLOBALS

---

▪ Syntax Explanation .....	42
▪ List of Parameters .....	42
▪ Interaction with SET GLOBALS and Other Statements .....	43

`GLOBALS [parameter=value ...]`

This command is used to set Natural session parameters.



**Note:** In batch mode, this command is only executable, if parameters are specified. For example, `GLOBALS SM=ON` can be executed in batch mode.

## Syntax Explanation

---

<b>GLOBALS</b>	If the GLOBALS command is entered without parameters, a screen appears where you can modify the parameter settings.
<i>parameter</i>	<p>Parameter settings can be supplied in any order and must be separated by a blank.</p> <p>If more parameters are specified than will fit on one command line, several GLOBALS commands must be issued.</p> <p>Example:</p> <pre>GLOBALS DC=, ID=.</pre> <p><b>Note:</b> Certain session parameters can be modified only using the above mentioned screen, but not via the command line.</p>

## List of Parameters

---

The following table contains a list of session parameters that can be specified with the system command GLOBALS.

Parameters	Function
CC	Error Processing in Batch Mode
CF	Character for Terminal Commands
CO	Compiler Output
CPCVERR	Code Page Conversion Error
DBSHORT	Interpretation of Database Short Names
DC	Character for Decimal Point Notation
DO	Display Order of Output Data
DU	Dump Generation
EJ	Page Eject
ENDIAN	Endian Mode for Compiled Objects



Parameters	Function
FS	Default Format/Length Setting for User-Defined Variables
GFID	Global Format IDs
IA	Input Assign Character
ID	Input Delimiter Character
IM	Input Mode
LE	Reaction when Limit for Processing Loop Exceeded
LS	Line Size
LT	Limit for Processing Loops
ML	Position of Message Line
PD	Size of Page Dataset
PM	Print Mode
PS	Page Size for Natural Reports
SA	Sound Terminal Alarm
SF	Spacing Factor
SM	Programming in Structured Mode
THSEPCH	Thousands Separator Character
ZD	Zero-Division Check
ZP	Zero Printing

## Interaction with SET GLOBALS and Other Statements

### Statement SET GLOBALS

The system command GLOBALS and the statement SET GLOBALS offer the same parameters for modification. They can both be used in the same Natural session. Parameter values specified with a GLOBALS command remain in effect until they are overridden by a new GLOBALS command or SET GLOBALS statement, the session is terminated, or you log on to another library.

### Other Statements Influencing the Session Parameter Settings

Some parameter values may be overridden during program execution using the LIMIT, EJECT, and FORMAT statements and by format entries specified in INPUT, DISPLAY, PRINT, and WRITE statements.



# 15 HELP

---

{ HELP } [ [NAT]nnnn ]
{ ? } [ USER[nnnn] ]

This command is used to invoke online help for error messages.



**Note:** This command is not executable in batch mode.

<b>HELP</b>	Displays the help menu.
<b>HELP [NAT]nnnn</b>	Entering <b>HELP</b> and a number (up to 4 digits, optionally prefixed by "NAT") displays the detailed message text for the Natural error condition associated with that number, that is, the long text of the Natural system error message <b>NATnnnn</b> .
<b>HELP USERnnnn</b>	Displays the long text of the library-specific error message number <i>nnnn</i> in the current library.



# 16 INPL

---

INPL [R]

This command is used to invoke the Natural INPL utility. This utility is *only* used for the loading of Software AG installation datasets into the system files as described in the INPL online help and in the platform-specific installation documentation.

Apart from that, you use the Object Handler to load objects into the system files.

<b>INPL</b>	If you enter the INPL command without any parameters, the INPL utility will be invoked.
<b>INPL R</b>	Invokes the INPL utility function Natural Security Recover which is only available if Natural Security is installed.  It can be used to reset the access to the Natural Security library SYSSEC to its original status. Then the user DBA, the library SYSSEC, and the link between the two will be redefined as after the initial installation, while all other links to SYSSEC will be cancelled. See also <i>Inaccessible Security Profiles</i> in the section <i>Countersignatures</i> of the <i>Natural Security</i> documentation.

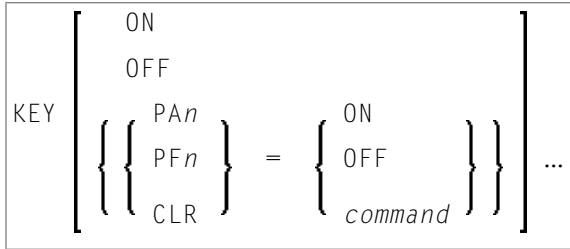
For further information, see *INPL Utility* in the *Tools and Utilities* documentation.



# 17 KEY

---

▪ Assigning Commands .....	50
▪ Activating/Deactivating All Keys - KEY ON/OFF .....	51
▪ Activating/Deactivating Individual Keys - KEY key=ON/OFF .....	51



This command is used to assign functions to keys on the keyboard of video terminals. Moreover, you can change, activate and deactivate the assigned functions.

This is possible for the following keys:

- PA1 to PA3,
- PF1 to PF24
- CLEAR

To each of these keys, you can assign one of the following functions:

- a Natural system command,
- a Natural terminal command,
- a user-defined command.

Natural will execute the assigned command whenever you press the corresponding key in command mode (**Direct Command** window).



#### Notes:

1. Assignments made with the system command KEY are totally independent of assignments made with a SET KEY statement in a program.
2. Function-key assignments can also be made by the Natural administrator via the profile parameter KEY.
3. This command is not executable in batch mode.

## Assigning Commands

If you enter only the command KEY (without parameters), the **Function-Key Assignments** screen will be displayed. On this screen, you can assign commands to the individual keys by entering the command names in the input fields.

To assign a different command to a key, you overwrite the existing entry in the input field.

To delete a command assignment, you delete the entry in the input field or overwrite it with blanks.



You can also assign commands to individual keys by specifying them directly with the `KEY` command. For example:

```
KEY PF1=CLEAR
```

If the assigned command contains blanks, it has to be enclosed in apostrophes. For example:

```
PF13='UPDATE OFF'
```

## Activating/Deactivating All Keys - KEY ON/OFF

With the command `KEY OFF/ON`, you deactivate/re-activate all function-key assignments:

<b>KEY OFF</b>	If you the press a function key, Natural will return an appropriate message indicating that the key is not active.
<b>KEY ON</b>	Re-activates all function-key assignments that have previously been deactivated with <code>KEY OFF</code> .

You can also activate/deactivate the keys by overwriting the entry `ON/OFF` in the field **Activate Keys** at the top right-hand corner of the **Function-Key Assignments** screen.

## Activating/Deactivating Individual Keys - KEY key=ON/OFF

With the command `KEY key=OFF/ON`, you deactivate/re-activate the command assigned to a specific *key*.

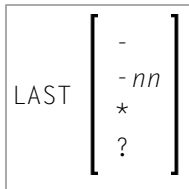
<b>KEY key=OFF</b>	Deactivates the command assigned to a specific <i>key</i> . For example: <pre>KEY PF24=OFF</pre>
<b>KEY key=ON</b>	Re-activates a previously deactivated command assignment. For example: <pre>KEY PF24=ON</pre>



**Note:** When you deactivated an individual key (for example, `PF24=OFF`), then deactivated all keys (`KEY=OFF`) and then activate all keys again (`KEY=ON`), the individually deactivated key is activated, too.



# 18 LAST



This command is used to display the system command(s) that was/were last executed. Moreover, you can have the displayed command(s) executed again. You can also overwrite them before they are executed.

Only system commands that you actually entered can be displayed via the `LAST` command; commands issued internally by Natural as a result of a command you entered are not available via `LAST`.

<b>LAST</b>	The command that was issued last is placed in the <b>Direct Command</b> window and can be executed.
<b>LAST -</b>	The command that was issued last is placed in the <b>Direct Command</b> window and can be executed.  If you enter <code>LAST -</code> again, the last but one command is placed in the <b>Direct Command</b> window.  By repeatedly entering <code>LAST -</code> , you can thus “page” backwards command by command.  <b>Note:</b> Instead of repeatedly entering it by hand, you can assign <code>LAST -</code> to a PF key via the system command <code>KEY</code> .
<b>LAST -nn</b>	Natural “remembers” up to the last 20 commands that were issued; <code>nn</code> must therefore not be greater than 20.  The last command but <code>nn</code> is placed in the <b>Direct Command</b> window and can be executed.
<b>LAST *</b>	A window is displayed showing the last 20 commands that were issued. Use PF8 and PF7 to scroll forward and backward if more than 10 commands are displayed:  ■ To execute a <i>single</i> command again, either mark the command with the cursor and press F5, or mark the command with any character and press ENTER.

	<ul style="list-style-type: none"><li>■ To execute <i>several</i> commands again, mark them with numbers in the order in which you wish them to be executed and press ENTER, the commands will then be executed in ascending order of numbers.</li></ul>
<b>LAST ?</b>	The Help function for the LAST command is called.

# 19 LASTMSG

---

## LASTMSG

This command is used to display additional information about the error situation which has occurred last.

When Natural displays an error message, it may in some cases be that this error is not the actual error, but an error caused by another error (which in turn may have been caused by yet another error, etc.). In such cases, the `LASTMSG` command allows you to trace the issued error back to the error which has originally caused the error situation.

When you enter the command `LASTMSG`, you will get - for the error situation that has occurred last - the error message that has been displayed, as well as all preceding (not displayed) error messages that have led to this error.

### ▶ To display information on the corresponding error

- Mark one of these messages with the cursor and press `ENTER`.

The following is displayed:

- error number;
- number of the line in which the error occurred;
- name, type and level of the object that caused the error;
- name, database ID and file number of the library containing the object;
- error class (system = error issued by Natural; user = error issued by user application);
- error type (runtime, syntax, command execution, session termination, program termination, remote procedure call);
- date and time of the error.



**Note:** The library SYSEXT contains a user application programming interface USR2006 which enables you to display in your Natural application the error information supplied by LASTMSG.

**Natural Remote Procedure Call (RPC):**


If an error occurs on the server, the following error information is not displayed: database ID, file number, date and time.

# 20 LIST

---

▪ Syntax Overview .....	58
▪ Displaying an Individual Source .....	59
▪ Displaying a List of Objects .....	60
▪ Displaying Directory Information .....	61
▪ Displaying Views .....	62
▪ Displaying File Information of Resource Objects .....	62
▪ Displaying File Information of Error Message Containers .....	62

This system command is used to display the source code of a single object or to list one or more objects which are contained in the current library.

 **Note:** This command is not executable in batch mode.

This chapter covers the following topics:

See also the descriptions of the commands [LIST XREF](#) and [LIST COUNT](#).

## Syntax Overview

---



**object-type**



```

*
{
  CLASS
  4
}
COPYCODE
DATA-AREAS
  GLOBAL
  LOCAL
  PARAMETER
  {
    DIALOG
    3
  }
  7 (for function)
  8 (for adapter)
MAP
  {
    PROCESSOR
    CP
    5
  }
PROGRAM
ROUTINES
  HELPROUTINE
  {
    SUBPROGRAM
    N
  }
SUBROUTINE
TEXT

```

### object-name

In place of *object-name*, you may specify the name of an object (8 characters long at maximum). You may also specify asterisk notation (\*), see the [examples](#) below.

## Displaying an Individual Source

<b>LIST</b>	If you enter only the LIST command itself, without any parameters, the contents of the source of the object currently selected will be listed.
<b>LIST</b> <i>object-name</i>	If you enter a single object name with the LIST command, you need not specify the <i>object-type</i> .
<b>LIST</b> <i>object-type object-name</i>	If you specify an <i>object-type</i> , you must also specify an <i>object-name</i> .  In both cases, the object's source code will be listed.

## Displaying a List of Objects

<b>LIST</b> <i>object-name</i>	To have all objects in the current library listed, except DDMs, you specify an asterisk (*) for the <i>object-name</i> , but no <i>object-type</i> .
<b>LIST</b> <i>object-type object-name</i>	To have all objects of a certain type listed, you specify a certain <i>object-type</i> and an asterisk (*) for the <i>object-name</i> .  If you wish a certain range of objects to be listed, you can use asterisk notation (*) for the <i>object-name</i> and/or wildcard notation (?).

### Examples

- List all objects in the current library, except views, resources, errors - regardless of their types:

```
LIST *
```

- List all subroutines in the current library:

```
LIST S *
```

- List all objects (of any type) whose names begin with SYS:

```
LIST SYS*
```

- List all maps whose names begin with SYS:

```
LIST M SYS*
```

- List directory information of object PRG01 in current library:

```
LIST DIR PRG01
```

- List all objects such as NATAL, NATURAL, NAT $vr$ AL (where  $vr$  stands for the relevant version and release numbers):

```
LIST N?T*AL
```

## Performing a Function on an Object

To perform a function on an object from the selection list, you simply mark the object with the appropriate function code in the left-hand column.

The function codes are:

Code	Function
C	Check the object's source code.
D	Read the object's source code.
E	Edit the object's source (equivalent to the system command <code>EDIT</code> ).
L	List the object's source code.
I	List directory information of the object's source code (equivalent to <code>LIST DIRECTORY object-name</code> ).
H	Print hardcopy of the object's source.
R	Run (that is, compile and execute) the object's source (equivalent to the system command <code>RUN</code> ).
X	Execute the object (equivalent to the system command <code>EXECUTE</code> ).
S	Stow the object in source and object form (equivalent to the system command <code>STOW</code> ).
U	Delete the object's source and object form.
.	End.

Enter a question mark (?) or use F2 to display the list of the available function codes for the selected object.

## Displaying Directory Information

<b>LIST DIRECTORY</b>	<p>Displays the directory information about the last active object currently in the source work area:</p> <ul style="list-style-type: none"> <li>■ <b>Source code:</b> "Saved-on" date and time, library name, user ID, programming mode (reporting or structured), Natural version, code page information (if available), operating system, size, encoding.</li> <li>■ <b>Object code:</b> "Cataloged-on" date and time, library name, user ID, programming mode, Natural version, code page information (if available), operating system/version, size, Endian mode.</li> </ul> <p>Directory information on the saved source code cannot be always exact, because the source code can be edited with non-Natural editors which are not under the control of Natural.</p>
-----------------------	--

<b>LIST DIRECTORY</b> <i>object-name</i>	Displays the directory information about the specified object. If you use asterisk notation (*) for <i>object-name</i> , the directory information of the existing objects is displayed sequentially.
---	---



**Note:** The code page information displayed shows the first 32 characters of the code page only.

## Displaying Views

<b>LIST VIEW</b>	Displays a list of all views (DDMs).
<b>LIST VIEW</b> <i>view-name</i>	If you specify a single view name, the specified view will be displayed.  For the <i>view-name</i> , you can use asterisk notation to display a list of all views (*) or a certain range of views (for example: A*).

## Displaying File Information of Resource Objects

<b>LIST RESOURCE</b> <i>name</i>	Displays the file information about the specified resource object. For <i>name</i> , you may only use asterisk notation (*).
----------------------------------	--

Example - Display the file information of all resource objects whose name starts with a W:

```
LIST RESOURCE W*
```

## Displaying File Information of Error Message Containers

<b>LIST ERROR</b> <i>name</i>	Displays the file information about the specified error message container <i>Nnn</i> APMSL.MSG, where <i>nn</i> is the language code. For <i>name</i> , you may only use asterisk notation (*).
-------------------------------	---

# 21 LIST COUNT

---



This command is used to list the number of Natural objects in your current library.

<b>LIST COUNT</b>	Displays the total number of objects.
<b>LIST COUNT *</b>	Displays the number of objects broken down by object types.
<b>LIST COUNT <i>name</i>&lt;</b>	Displays the number of objects whose names are less/equal <i>name</i> .
<b>LIST COUNT <i>name</i>&gt;</b>	Displays the number of objects whose names are greater/equal <i>name</i> .
<b>LIST COUNT <i>name</i>*</b>	Displays the number of only those objects whose names begin with <i>name</i> .



**Note:** If there are objects listed under object type `undefined`, this indicates that the library contains objects whose version is not compatible.



## 22 LIST XREF

---

LIST XREF

This command is only available if Predict has been installed.

It is used to display all active cross-reference data for the current library.

For further information, see *List XREF For Natural* in the Predict documentation.

---



# 23 LOGOFF

---

LOGOFF

Related command: [LOGON](#).

This command is used to cause the library ID to be set to SYSTEM and the Adabas password to be set to blanks. The contents of the source program work area are not affected by this command.

LOGOFF has no effect on Natural global parameter settings.

For information on LOGOFF processing under Natural Security, see *How to End a Natural Session* in section *Logging On* of the *Natural Security* documentation.



**Note:** LOGOFF does *not* cause the Natural session to be terminated.

## ▶ To terminate the session

- Use the system command [FIN](#), or execute a program that contains a TERMINATE statement.



# 24 LOGON

`LOGON library-id [password]`

Related command: [LOGOFF](#).

This command is used to log on to a library in your environment or create a new library. In the specified library, all newly created source or object programs saved during the session will be stored (unless you explicitly specify another library ID in a [SAVE](#), [CATALOG](#) or [STOW](#) command).

The LOGON command has no direct effect on the source program in the currently active window.

LOGON causes all Natural global data areas and application independent variables (AIVs), all assignments made using the SET KEY statement and retained ISN lists to be released. Data definition modules (DDMs) contained in the DDM buffer area are also released.

<b>LOGON</b> <i>library-id</i>	The library ID can be 1 to 8 characters long and must not contain blanks. It can consist of the following characters:	
	A - Z	upper-case alphabetical characters
	0 - 9	numeric characters
	-	hyphen
	_	underscore
	The first character of a library ID must be an upper-case alphabetical character.	
<b>LOGON</b> <i>library-id</i> <i>password</i>	The Adabas password; see <i>Session Parameters</i> in section <i>Library Maintenance</i> of the <i>Natural Security</i> documentation.	

For information on LOGON processing under Natural Security, see *Logging On* in the *Natural Security* documentation.



# 25 MAIL

---

```
MAIL [ { *  
        ?  
        mailbox-id } ]
```

This command is used to invoke a mailbox which is a kind of “notice board” used to broadcast messages under Natural Security. The contents and/or expiration date of the mailbox can be modified.

<b>MAIL</b>	If you enter the <code>MAIL</code> command without any parameters, a window is displayed prompting you to enter a mailbox ID.
<b>MAIL *</b>	A list of all mailboxes you may use is displayed, and you may then select a mailbox from the list.
<b>MAIL ?</b>	
<b>MAIL mailbox-id</b>	If you specify a <i>mailbox-id</i> (maximum 8 characters), the corresponding mailbox is invoked directly. The <i>mailbox-id</i> must have been defined in Natural Security.

For further information, see *Mailboxes* in the *Natural Security* documentation.



# 26 PROFILE

---

This command is available only if Natural Security is installed.

PROFILE

This command is used to display the security profile currently in effect. This profile informs you of the conditions of use in effect for you in your current Natural environment.

For further information, see *PROFILE Command* in the *Natural Security* documentation.







# 27 PURGE

---

PURGE [*object-name* ...]

This command is used to delete one or more source objects.

-  **Note:** If the Natural profile parameter `RECAT` is set to `ON`, the `PURGE` command will be rejected for a source for which a corresponding cataloged object exists.
-  **Note:** If DDMs reside in a library of the `FUSER` system file, they can be deleted with the `PURGE` command.

<b>PURGE</b>	If you enter the <code>PURGE</code> command without an <i>object-name</i> , a list of all objects in the current library will be displayed; on the list, you can then mark the objects to be deleted.
<b>PURGE</b> <i>object-name</i>	As <i>object-name</i> , you specify the name(s) of the object(s) to be deleted. You can only delete objects that are stored in the library to which you are currently logged on.  If you wish to delete all objects whose names begin with a specific string of characters, use asterisk notation (*) for the <i>object-name</i> .



# 28 READ

---

`READ object-name [library-id]`

Related command: [EDIT](#).

This command is used to transfer an object that is stored in source form into the source work area. Any object currently in the source work area will be overwritten by the object read.

See also *Object Naming Conventions* in the *Using Natural* documentation.

<code>READ <i>object-name</i></code>	The name of the object to be read.  If <i>object-name</i> is specified without a library ID, the object will be read only if it is stored in the library to which you are currently logged on.
<code>READ <i>object-name</i> <i>library-id</i></code>	The library in which the object to be read is contained.  If both <i>object-name</i> and <i>library-id</i> are specified, Natural will only read the object if it is stored under the specified library ID.



# 29 RENAME

---

```
RENAME [old-name [new-name [new-type]]
```

This command is used to give a Natural programming object another name. In addition, you can change the object type.

You can only rename one object at a time. The object to be renamed must be stored in the library to which you are currently logged on. To ensure consistency, Natural will rename source code or object module or both.

See also *Object Naming Conventions* in the *Using Natural* documentation.

RENAME	If you issue the command without parameters, a <b>Rename Object</b> window appears where you can specify the same parameters as in the command line.	
<i>old-name</i>	As <i>old-name</i> you specify the existing name of the object to be renamed.	
<i>new-name</i>	As <i>new-name</i> you specify the name under which the object is to be stored from now on.	
<i>new-type</i>	When you rename an object in source form, you can also change its object type by specifying the corresponding character for <i>new-type</i> .  The possible values you can specify for <i>new-type</i> are:	
	4	Class
	5	Processor
	7	Function
	8	Adapter
	9	Resource
	A	Parameter data area
	C	Copycode
	G	Global data area
	H	Helproutine

## RENAME

---

L	Local data area
M	Map
N	Subprogram
P	Program
S	Subroutine
T	Text
Y	Rule
Z	Recording

# 30 RENUMBER

---

RENUMBER [(n)]

This command is used to renumber the lines in the source program currently in the source work area.

RENUMBER	If you enter the command without parameter, the increment to be used for renumbering is 10.
RENUMBER (n)	n can be used to specify a value between 1 and 10 as increment for renumbering.





# 31 RETURN

```
RETURN [ { I }
        { nn }
        { * } ]
```

This command is used to return to a previous (or initial) Natural application.

RETURN	<p>If RETURN is specified without any parameters, control will be returned to the previous application (as defined with the system command <a href="#">SETUP</a>). All information about this previous application will be deleted. If no previous application exists, control is returned to the initial application.</p> <p>If RETURN is issued and no return point is set, the RETURN command will be ignored.</p> <p><b>Under Natural Security:</b></p> <p>A LOGOFF command will be executed if RETURN is issued and no return point has been set.</p>
RETURN I	This command causes control to be returned directly to the initial application. This option also causes Natural to delete all definitions of previous applications (except that of the initial application).
RETURN nn	This command causes control to be returned to the <i>nn</i> th previous application. When this option is used, all information for applications subsequent to the <i>nn</i> th application is deleted.
RETURN *	This command will display a list of all return points which are currently set up. On the list you may then select the return point to which you wish to return.

See the [SETUP](#) command for further information and examples.



# 32

## RPCERR

---

RPCERR

This command is used to display the last Natural error number and message if it was RPC related, and it also displays the last Broker reason code and associated message. Additionally, the node and server name from the last Broker call can be retrieved.

For further information, see *Monitoring the Status of an RPC Session* in the *Operating a Natural RPC Environment* section of the *Natural Remote Procedure Call (RPC)* documentation.



# 33 RUN

```
RUN [REPEAT] [program-name [library-id]]
```

This command is used to compile and execute a source program. The program may be in the source work area or in the Natural system file.

RUN	If <i>program name</i> is not specified, Natural will compile and execute the program currently residing in the work area.
REPEAT	REPEAT defines that if the program being executed produces multiple screen output, the screens are to be output one after another without intervening prompting messages. When the program terminates, Natural will enter command mode.
<i>program-name</i>	The name of the program to be run.  If <i>program-name</i> is specified without a library ID, Natural will read the source program into the source work area, compile, and execute the specified program only if it is stored under the current library ID. If it is not stored under the current library ID, an error message will be issued.
<i>library-id</i>	The library in which the program to be run is contained.  If both <i>program-name</i> and <i>library-id</i> are specified, Natural will retrieve, compile, and execute the specified program only if it is stored under the library ID specified. If it is not stored under the current library ID, an error message will be issued.  The setting for <i>library-id</i> must not begin with SYS (except SYSTEM).



# 34 SAVE

```
SAVE [object-name [library-id]]
```

Related commands: [STOW](#) | [CATALOG](#).

This command is used to save the source code of the programming object currently in the work area of the editor and store it as a source object in the Natural system file.



**Caution:** The `SAVE` command cannot be used if the profile parameter `RECAT` has been set to `ON`; in this case, use the [STOW](#) command to compile and store the object.

<b>SAVE</b>	If you use the command without <i>object-name</i> , the current source object in the source work area will be saved in the current library. An existing source code will be replaced.
<b>SAVE</b> <i>object-name</i>	A new source object is created. As <i>object-name</i> , you specify the name under which the source object is to be saved. The new source object is stored in the current library. If the source object exists, the command is rejected.
<b>SAVE</b> <i>object-name</i> <i>library-id</i>	When you save a source object under a different name or save a newly created object, the source object will, by default, be stored in the current library. If you wish to store it in another library, you have to specify the desired <i>library-id</i> after the <i>object-name</i> . A new source object is created, if the source object exists, the command is rejected.





# 35 SCAN


---


▪ Menu Options .....	92
▪ Subcommands .....	93
▪ SCAN under Natural Security .....	94

## SCAN

This command is used to search for a string of characters within an object, with the option to replace the string with another string.

The object may be a single object, all objects beginning with a specified setting, or all objects within a library. The scan may also be restricted to a specific object type.

 **Important:** The source work area is used by the SCAN command. Therefore, a [SAVE](#) or [STOW](#) command should be issued before using the SCAN command.

 **Note:** This command is not executable in batch mode.

This chapter covers the following topics:

## Menu Options

Any objects except maps, DDMs and data areas can be modified using the scan utility. You can also use the full screen editor to modify other lines than scanned lines. When the object is modified, save it and close the editor. You can then continue scan processing.

When you issue the SCAN command, a window is displayed where you can specify the following:

Field	Explanation	
Scan value	The value to be searched for.	
Replace value	The value which is to replace the scan value.	
Delete value	Will cause the scan value to be deleted (Y/N).	
Object name	The object(s) to be scanned:	
	<i>blank</i>	All objects.
	*	
	<i>name*</i>	All objects whose names begin with <i>name</i> .
Object type	The type(s) of object to be scanned:	
	P	Programs
	C	Copycodes
	N	Subprograms
	S	Subroutines
	H	Help routines
	M	Maps
G	Global data areas	



Then you can select one of the following commands:

Command	Function
Edit	Edit object.
List	List object as it currently appears in the source work area.
Ignore	Ignore the object currently being scanned, do not save any modifications, and continue with next object.
Quit	Terminate scan processing.

A subcommand can also be invoked by entering its first character.

To deactivate command mode, press ESC.

### Editing Rules

- If the Replace option is used and/or an object is updated in the scan utility, the object will always be saved unless **Ignore** or **Quit** is specified before the next object is scanned.
- Lines containing `PASSWORD=`, `PASSW=`, `CIPHER=`, or `CIPH=` will be ignored by the SCAN command.
- The line length of the source object in the scan utility is limited to 72.
- If the replace value causes a line to exceed 80 characters, the line will be split automatically.

## SCAN under Natural Security

---

In order to use SCAN in a Natural Security environment, the system commands `LIST`, `EDIT`, and `READ` must be allowed in the current library's security profile. If the Replace option is to be used, the system command `SAVE` must also be allowed.

Under Natural Security, the use of the SCAN command may be disallowed in some libraries.

# 36 SCRATCH

---

```
SCRATCH [ { *  
          object-name... } ]
```

This command is used to delete one or more objects - in both source and object form. The contents of the source work area is not affected.

<b>SCRATCH</b>	If you enter the SCRATCH command without an <i>object-name</i> or without an
<b>SCRATCH *</b>	<i>object-name</i> but with an asterisk (*), a list of all objects or all selected objects in the current library will be displayed. On the list you may then mark the objects to be deleted.
<b>SCRATCH <i>object-name</i></b>	As <i>object-name</i> , you specify the name(s) of the object(s) to be deleted. You can only delete objects which are stored in your current library.  If you wish to delete all objects whose names begin with a specific string of characters, use asterisk notation (*) for the <i>object-name</i> .



**Note:** If an FDIC system file is specified in the parameter file which is not valid, Natural will display an appropriate error message when the SCRATCH command is issued.



# 37 SETUP

---

▪ Syntax Explanation .....	98
▪ SETUP/RETURN Example .....	99

```
SETUP [application-name] [command-name] [I]
```

This command is used to define applications to which control is to be returned using the [RETURN](#) command. This allows you to easily transfer from one application to another during a Natural session.

This chapter covers the following topics:

## Syntax Explanation

---

The command syntax and the parameters that can be issued with the `SETUP` system command are explained below. If a parameter is to be omitted, you may use the input delimiter character to mark the beginning of the following parameter(s).

<b>SETUP</b>	If <code>SETUP</code> is issued without parameters, a menu will be displayed for the purpose of entering the command information.
<i>application-name</i>	<p>The name of the application to which control is to be returned. A maximum of 8 characters may be used (A8).</p> <p>If <i>application-name</i> is blank, a <a href="#">LOGON</a> command will not be issued. This permits multiple return points within the same application.</p> <p>If <i>application-name</i> is "*", the current setting of the system variable *LIBRARY-ID (that is, at the time <code>SETUP</code> is issued) is used to create the <code>LOGON</code> command when <code>RETURN</code> is issued.</p>
<i>command-name</i>	<p>The name of the command which is to be executed when control is returned to the application. A maximum of 60 characters may be used (A60).</p> <p>If <i>command-name</i> is blank, no command will be issued after the <code>LOGON</code>. This is useful for applications under Natural Security for which a startup program has already been defined.</p> <p>If <i>command-name</i> is "*", the current setting of the system variable *STARTUP (that is, at the time <code>SETUP</code> is issued) is used as the startup command when <code>RETURN</code> is issued.</p>
I	<p>If the I option is specified, all return points defined with previous <code>SETUP</code> commands will be deleted and the application specified with <code>SETUP I</code> will be defined as the new initial application.</p> <p>In a non-Security environment, if you log on from library <code>SYSTEM</code> to another library and no return point has been set, this other library will automatically be set as initial return point.</p>



---

## SETUP/RETURN Example

---

1. User starts Natural session (default application is APPL1).

Return point APPL1 is defined on Level 1.

2. User issues command LOGON APPL2.
3. User executes a program which stacks two commands (establish return point and go to another application):

```
SETUP *,MENU  
LOGON APPL3
```

Return point APPL2, STARTUP MENU is defined on Level 2.

4. User issues command LOGON APPL4 (user selects another application).
5. User presses a PF key which has the setting RETURN. Natural will issue for the user:

```
LOGON APPL2  
MENU
```

Return to APPL2, delete Level 2.

6. User executes a program which stacks:

```
SETUP *,MENU  
LOGON APPL5
```

Return point APPL2, STARTUP MENU is defined on Level 2.

7. User executes a program which stacks:

```
SETUP *,MENU  
LOGON APPL6
```

Return point APPL5, STARTUP MENU is defined on Level 3.

8. User executes a program which stacks:

```
SETUP *,MENU  
LOGON APPL7
```

Return point APPL6, STARTUP MENU is defined on Level 4.

9. User executes a program which stacks:

```
SETUP *,MENU  
LOGON APPL8
```

Return point APPL7, STARTUP MENU is defined on Level 5.

10. User executes a program which stacks:

```
SETUP *,MENU  
LOGON APPL9
```

Return point APPL8, STARTUP MENU is defined on Level 6.

11. User issues command RETURN 2 (return two levels back).

Natural will return user to APPL7, since that was the second previous session (all information for APPL8 is now lost). Level 6 (APPL8) is deleted, Level 5 (APPL7) is activated and level deleted.

12. User issues command RETURN.

Level 4 (APPL6) is activated, level deleted. Natural will return user to APPL6, since that was the session previous to APPL7.

13. User issues command RETURN.

Level 3 (APPL5) is activated, level deleted. Natural will return user to APPL5, since that was the session previous to APPL6.

14. User issues command RETURN 1.

Level 2 (APPL2) is deleted, Level 1 (APPL1) is activated.

# 38 STOW

---

```
STOW [object-name [library-id]]
```

Related commands: [SAVE](#) | [CATALOG](#).

This command is used to compile and store a Natural programming object (in both source and object form) in the Natural system file. You can regard this command as a `CATALOG` followed by a `SAVE`.

<b>STOW</b>	If you use the command without <i>object-name</i> , the source code held in the source area as well as the generated code will be stored under the same name in the current library. Existing source and object code will be replaced.
<b>STOW</b> <i>object-name</i>	Use this command syntax to store a new object (source and generated code) named <i>object-name</i> in the current library. If the object exists in either source or cataloged form, the command is rejected.
<b>STOW</b> <i>object-name</i> <i>library-id</i>	If both <i>object-name</i> and <i>library-id</i> are specified, a new object will be created and stored under that name in the specified library ID. If the object exists in either source or cataloged form, the command is rejected.



**Note:** If an FDIC system file is specified in the parameter file which is not valid, Natural will display an appropriate error message when the `STOW` command is issued.



# 39 STRUCT

---

- Indentation of Source Code Lines ..... 104

STRUCT [(n)]

This command is used to perform structural indentation of the source code of the programming object currently in the work area of the editor.

<b>STRUCT</b>	By default (that is, if "(n)" is not specified), indentation is by 2 positions.
<b>STRUCT (n)</b>	The parameter "(n)" may be supplied to specify the number of spaces used for indentation. Possible values: 1 - 9. Example: <pre>STRUCT (5)</pre>

The following types of statements are affected by the STRUCT command:

- processing loops (READ, FIND, FOR, etc.),
- conditional statement blocks (AT BREAK, IF, DECIDE FOR, etc.),
- DO/DOEND statement blocks,
- DEFINE DATA blocks,
- inline subroutines.

This chapter covers the following topics:

## Indentation of Source Code Lines

You can have a source program indented so that the indentation of source-code lines reflects the structure of the program.



**Note:** Indentation is performed differently for a reporting-mode program than for a structured-mode program.

### Partial Indentation

You can exclude sections of your program source from structural indentation by using the special statements `/*STRUCT OFF` and `/*STRUCT ON`. These must be entered at the beginning of a source-code line. The source-code lines between these two statements will remain as they are when you issue the STRUCT command.

## Example of Structural Indentation

Program before being structurally indented:

```
DEFINE DATA LOCAL
1 EMPL VIEW OF EMPLOYEES
2 PERSONNEL-ID
2 FULL-NAME
3 FIRST-NAME
3 NAME
1 VEHI VIEW OF VEHICLES
2 PERSONNEL-ID
2 MAKE
END-DEFINE
FIND EMPL WITH NAME = 'ADKINSON'
IF NO RECORDS FOUND
WRITE 'NO RECORD FOUND'
END-NOREC
FIND (1) VEHI WITH PERSONNEL-ID = EMPL.PERSONNEL-ID
DISPLAY EMPL.PERSONNEL-ID FULL-NAME MAKE
END-FIND
END-FIND
END
```

The same program after being structurally indented:

```
DEFINE DATA LOCAL
1 EMPL VIEW OF EMPLOYEES
  2 PERSONNEL-ID
  2 FULL-NAME
    3 FIRST-NAME
    3 NAME
1 VEHI VIEW OF VEHICLES
  2 PERSONNEL-ID
  2 MAKE
END-DEFINE
FIND EMPL WITH NAME = 'ADKINSON'
  IF NO RECORDS FOUND
    WRITE 'NO RECORD FOUND'
  END-NOREC
  FIND (1) VEHI WITH PERSONNEL-ID = EMPL.PERSONNEL-ID
    DISPLAY EMPL.PERSONNEL-ID FULL-NAME MAKE
  END-FIND
END-FIND
END
```





# 40

## SYSDDM

---

SYSDDM

This command is used to invoke the DDM Services which offer functions that are needed to create and maintain Natural data definition modules (DDMs).

For further information, see *DDM Services* in the *Editors* documentation.



**Note:** This command is not executable in batch mode.

### **Note Concerning Natural Single Point of Development:**

This command is not available via Natural Studio's command line in a remote development environment, because DDMs are listed in the tree view under the node DDM and all functions of the SYSDDM utility are available via the context menu or the menu bar.



# 41

## SYSERR

---

SYSERR

This command is used to invoke the `SYSERR` utility.

With the `SYSERR` utility, you can write your own application-specific messages.

- You can use the `SYSERR` utility to separate error or information messages from your Natural code and manage them separately.
- As well as unifying messages and defining message ranges for different kinds of messages, you can translate messages into another language and attach a long text to a message.
- You can also use the `SYSERR` utility to modify the texts of existing Natural system messages, although this is not recommended as modifications will be lost with new Natural releases.

For further information, see *SYSERR Utility* in the *Tools and Utilities* documentation.



# 42 SYSEXT

---

SYSEXT

This command is used to invoke the SYSEXT utility.

This utility is used to display various Natural application programming interfaces contained in the library SYSEXT.

For further information, see *SYSEXT - Natural Application Programming Interfaces in the Tools and Utilities* documentation.



# 43 SYSEXV

---

SYSEXV

This command is used to invoke the SYSEXV utility with examples of the new features of the current Natural versions.

For further information, see *SYSEXV Utility* in the *Tools and Utilities* documentation.





# 44

## SYSFILE

---

### SYSFILE

This command is used to display work and print files information. You can obtain information about the following:

- reports,
- logical devices,
- defined physical devices,
- defined printer profiles, and
- defined workfiles.

For further information on work and print files, see

- *Printer Profiles* in the *Configuration Utility* documentation, and
- *Device/Report Assignments* in the *Configuration Utility* documentation,
- *Work Files* in the *Operations* documentation,



# 45 SYSMAIN

---

## SYSMAIN

This command is used to invoke the `SYSMAIN` utility. You use this utility to perform operations such as copy, move and delete on Natural objects. The `SYSMAIN` utility is also used to transfer objects within the Natural system from one environment to another using the import function.

For further information, see *SYSMAIN Utility* in the *Tools and Utilities* documentation.



**Note:** This command is not executable in batch mode.



# 46

## SYSNCP

---

SYSNCP

This command is used to invoke the `SYSNCP` utility.

For further information, see *SYSNCP Utility* in the *Tools and Utilities* documentation.



# 47

## SYSOBJH

---

SYSOBJH

This command is used to invoke the Object Handler. You use the Object Handler to process Natural and non-Natural objects for distribution in Natural environments.

For further information, see *Object Handler* in the *Tools and Utilities* documentation.





# 48 SYSPROD

---

## SYSPROD

This command is used to ascertain which products are installed at your Natural site. You are given information on your current Natural version, Natural selectable units and products running with or under Natural.

When you enter the command, a dialog displays information such as the following for each product installed:

- the product name,
- the product version (see also *Version* in the *Glossary*),
- the installation date and time.

For some of the products listed, you can get additional information by marking them with a line command in the **Cmd** column of the dialog.

Line Command	Description
EX	Display extended product information.
HI	Display history of product information.
SC	Display subcomponents of product.



**Note:** For some products, no line commands are allowed.



# 49 SYSPROF

---

SYSPROF

This command is used to display the current definitions of the Natural system files.

For each system file, the following information is displayed (this corresponds to pressing PF4 when one of the pages listed below is currently shown):

- the file name
- the database ID
- the file number
- the database type

In addition, the following information can be displayed for each defined combination of database ID and file number:

- the path in the file system (when you press PF5)
- the logical file number, if assigned (when you press PF6)



# 50

## SYSRPC

---

SYSRPC

This command is used to invoke the `SYSRPC` utility.

The `SYSRPC` utility provides functions for maintaining remote procedure calls.

For further information, see *SYSRPC Utility* in the *Tools and Utilities* documentation.

For information on how to apply the `SYSRPC` utility functions to establish a framework for communication between server and client systems, refer to the *Natural Remote Procedure Call (RPC)* documentation.



# 51 SYSWIZDB

---

`SYSWIZDB`

This command is used to invoke the Data Browser, a development tool wizard within Natural Studio. It enables you to display and print or store file structures.





# 52

## SYSWIZDW

---

SYSWIZDW

This command is used to invoke the Dialog Wizard, a tool for creating dialogs for specific purposes. The defined dialogs can have several layouts that adapt to desired requirements.



# 53 TECH

---

## TECH

This command is used to display the following technical and other information about your Natural session:

- user ID
- library ID
- Natural version, release and SM level
- startup transaction
- Natural Security indicator
- operating system name and version
- machine class
- hardware
- TP monitor (Mainframes and Windows (\*TPSYS) in remote configuration only)
- device type
- terminal ID (Mainframes and Windows in remote configuration only)
- code page
- locale
- last command issued
- information on the last error that occurred
- names, database IDs and file numbers of all currently active steplibs
- names, types and levels of the currently active programming object and all objects on higher levels, as well as the line numbers of the statements invoking the subordinate programming objects (Mainframes, UNIX and OpenVMS only).



### Notes:

1. For character-user-interface applications only: To display this information from any point in an application, you can use the terminal command `%<TECH`.
2. This command is also available in a remote session. All information can be read in batch mode.

# 54 UNCATALOG

---

UNCATALOG [*object-name* ...]

This command is used to delete one or more object modules.

You can only delete objects which are stored in the library to which you are currently logged on. The contents of the source work area is not affected by the UNCATALOG command.

<b>UNCAT</b>	If you enter the UNCATALOG command without an <i>object-name</i> or with an asterisk (*), a list of all cataloged objects in the current library will be displayed; on the list, you can then mark the object(s) to be deleted.
<b>UNCAT *</b>	
<b>UNCAT</b> <i>object name</i>	As <i>object-name</i> , you specify the name of the object to be deleted.  If more than one object is to be deleted, the <i>object-names</i> must be separated by one or more blanks (or the currently defined delimiter character).  If you wish to delete all objects whose names begin with a specific string of characters, use asterisk notation (*) for the <i>object-name</i> . A list containing all objects selected will be displayed. On the list, you can then mark the object(s) to be deleted.



**Note:** If an FDIC system file is specified in the parameter file which is not valid, Natural will display an appropriate error message when the UNCATALOG command is issued.

---

# 55 UPDATE

---

UPDATE	{ ON OFF }
--------	---------------

This command is used to prevent (or allow) database updating being carried out by a program.

<b>UPDATE ON</b>	This allows updating. This command will be ignored if the Natural administrator has made updating impossible during Natural installation.
<b>UPDATE OFF</b>	This prevents updating which would normally be performed as a result of an UPDATE, STORE, or DELETE statement. Programs containing these statements will execute normally but no modification of the database will occur. When an update operation is encountered, a message will be displayed instead of a database update being performed.

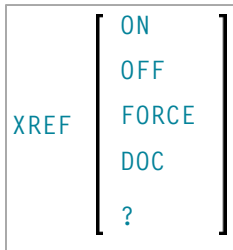
When the system command **CHECK** is used with UPDATE OFF, an error message is displayed. The UPDATE command has no effect on other Natural system commands.





# 56 XREF

---



This command is only available if Predict has been installed. It controls the usage of the Predict function "active cross-references".

The active cross-reference facility automatically creates documentation in the Data Dictionary about the objects with a program/data area reference. These objects include programs, subprograms, subroutines, help routines, maps, data areas, database views, database fields, user-defined variables, processing rules, error numbers, work files, printers, classes and retained ISN sets.

The active cross-reference is created when a program/data area is cataloged.

To look at cross-reference data, you use the XREF option of the system command [LIST](#).

For further information on active cross-references, see the Predict documentation.

The following command options are available:

<b>XREF</b>	If you enter the XREF command without parameters, a menu/dialog is displayed where you specify the desired option.
<b>XREF ON</b>	This command activates the active cross-reference function. Cross-reference data will be stored in the respective Predict entries each time a Natural program/data area is cataloged.
<b>XREF OFF</b>	This command deactivates the active cross-reference facility. No cross-reference data will be stored. Existing cross-reference data for the object being cataloged will be deleted.

<b>XREF FORCE</b>	The object can only be cataloged if a Predict entry exists for it. When the object is cataloged, its cross-reference data will be stored in Predict. If no Predict entry exists, the object cannot be cataloged.
<b>XREF DOC</b>	The object can only be cataloged if a Predict entry exists for it. However, when the object is cataloged, no cross-reference data will be stored in Predict, and existing cross-reference data for the object will be deleted. If no Predict entry exists, the object cannot be cataloged.
<b>XREF ?</b>	With XREF ? you can call the Help function for the XREF command.

### **Natural Security Considerations**

If Natural Security is installed, the setting for XREF may be set for each library in the library security profile. Depending on the security profile, some options of the XREF command may not be available to you.