Natural

System Commands

Version 9.2.5

October 2026

ADABAS & NATURAL

This document applies to Natural Version 9.2.5 and all subsequent releases.

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

Copyright © 1979-2026 Software GmbH, Darmstadt, Germany and/or its subsidiaries and/or its affiliates and/or their licensors.

The name Software AG and all Software GmbH product names are either trademarks or registered trademarks of Software GmbH and/or its subsidiaries and/or its affiliates and/or their licensors. Other company and product names mentioned herein may be trademarks of their respective owners.

Detailed information on trademarks and patents owned by Software GmbH and/or its subsidiaries is located at https://softwareag.com/licenses.

Use of this software is subject to adherence to Software GmbH's licensing conditions and terms. These terms are part of the product documentation, located at https://softwareag.com/licenses 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, license terms, additional rights or restrictions, please refer to "License Texts, Copyright Notices and Disclaimers of Third-Party Products". For certain specific third-party license restrictions, please refer to section E of the Legal Notices available under "License Terms and Conditions for Use of Software GmbH Products / Copyright and Trademark Notices of Software GmbH Products". These documents are part of the product documentation, located at https://softwareag.com/licenses and/or in the root installation directory of the licensed product(s).

Use, reproduction, transfer, publication or disclosure is prohibited except as specifically provided for in your License Agreement with Software GmbH.

Document ID: NATMF-NNATSYSCOM-925-20261009

Table of Contents

Preface ix
1 About this Documentation 1
    Document Conventions 2
    Online Information and Support 2
    Data Protection 3
2 Issuing System Commands 5
    Command Input 6
    Command Line 6
    NEXT Prompt 6
    MORE Prompt 7
3 System Command Syntax 9
    Syntax Elements 10
    Example of Command Syntax 11
4 System Commands Grouped by Category 13
    Navigating in Natural 14
    Natural Development Environment 14
    Managing Applications with Natural Objects 15
    Monitoring, Debugging and Tracing 16
    Natural with Other Products 17
5 AIV 19
6 BUS 21
7 CATALL 23
    Catalog Objects from/to 24
    Predict Set and Set User 25
    Select Cataloged Source Objects Only, or Select All Source Objects 25
    Select Object Types 25
    Select Function 26
    Select Options 26
    Selection List 27
    Direct Command Syntax 28
8 CATALOG 31
9 CHECK 33
10 CLEAR 35
11 COMPOPT 37
    COMPOPT Syntax Explanation 38
    Specifying Compiler Keyword Parameters 38
    General Compilation Options 39
12 CPINFO 59
13 DELETE 63
    DELETE Syntax Explanation 64
    Selection List 65
    Safeguard Against Accidental Deletion 65
    DELETE Examples 65

14 DUMP 67
15 EDIT 69
EDIT Syntax 1 70
EDIT Syntax 2 72
EDIT Syntax 3 72
16 EDT 73
EDT Syntax Explanation 74
EDT Subcommands 74
EDT Function Keys 75
17 EXECUTE 77
EXECUTE Syntax Explanation 78
Examples of EXECUTE Command 79
18 FIN 81
19 GLOBALS 83
GLOBALS Syntax Explanation 84
GLOBALS Parameters 84
Interaction with SET GLOBALS and Other Statements 85
20 HELP 87
21 INPL 89
22 KEY 91
Assigning Commands 92
Activating/Deactivating All Keys - KEY ON/OFF 93
Activating/Deactivating Individual Keys - KEY key=ON/OFF 93
23 LAST 95
24 LASTMSG 97
25 LIST 99
LIST Syntax Overview 100
Listing the Contents of the Work Area 107
Displaying an Individual Source Code 107
Displaying a List of Objects 108
Displaying a Presorted List of Preselected Objects 108
Displaying Numbers and Sizes of Objects 108
Displaying Sources Sequentially 109
Displaying Directory Information 109
Displaying Long Names of Cataloged Subroutines, Classes, and Functions 111
Displaying NOC Options of Cataloged Objects 111
Displaying Compiler Options of Cataloged Objects 111
Displaying Compiler Output of a Cataloged Object 111
Displaying DDMs (Views) 113
Options 113
List of Objects 117
List of Source 125
Defining an Individual List Profile 130
26 LISTDBRM 135
27 LISTPROF 139

28 LIST XREF 143
29 LISTSQL 145
30 LISTSQLB 149
31 LOGOFF 153
32 LOGON 155
33 MAIL 157
34 MAINMENU 159
35 NATQVS 161
36 NOCOPT 163
37 NOCSHOW 165
38 NOCSTAT 167
39 PROFILE 169
40 PROFILER 171
41 RDC 173
42 READ 175
43 RENAME 177
44 RENUMBER 179
45 RETURN 181
46 ROUTINES 183
47 RPCERR 185
48 RUN 187
49 SAVE 189
50 SCAN 191
SCAN Menu Options 192
SCAN Edit Commands 195
SCAN via Direct Command 195
Defining an Individual SCAN Profile 199
SCAN Under Natural Security 200
51 SCRATCH 201
52 SETUP 203
SETUP Syntax Explanation 204
SETUP/RETURN Example 205
53 SHOWDBS 207
Corresponding API USR8211N 209
54 SQLDIAG 211
55 SQLERR 215
56 STOW 217
57 STRUCT 219
Generate Structured Source into Work Area 220
Display Structure of Source 223
Print Structure of Source 224
Write Structure of Source into Work Area 224
58 SYSADA 227
59 SYSAPI 229
60 SYSBPM 231

61 SYSCP 233
62 SYSDB2 235
63 SYSDDM 237
64 SYSEDT 239
65 SYSERR 241
66 SYSEXT 243
67 SYSEXV 245
68 SYSFILE 247
69 SYSMAIN 249
70 SYSNCP 251
71 SYSOBJH 253
72 SYSPARM 255
73 SYSPROD 257
About SYSPROD 258
Using the Command in Online Mode 258
Using the Command with Additional Keywords (Recommended in Batch Mode) 259
Application Programming Interfaces 259
74 SYSPROF 261
75 SYSRPC 263
76 SYSTP 265
77 TECH 267
78 TEST 269
79 TEST DBLOG 271
80 UNCATALOG 273
81 UNLOCK 275
Unlocking Natural Objects 276
UNLOCK Parameter Descriptions 277
Parameter Processing and Display of Objects Found 278
Batch Processing 279
82 UPDATE 281
83 WHICH 283
Direct Command Syntax 284
WHICH Parameter Description 284
Default Output of WHICH Command 284
Full List of WHICH Command 285
Batch Processing 286
WHICH Command under Natural Security 286
84 XREF 287
85 ZIIP 289
Syntax of ziip-subfunction-and-option 290
ZIIP Syntax Element Description 291
zIIP Reports 293
ZIIP Menu for Report Output Options 297
User Exit for ZIIP PRINT 298

Application Programming Interface (API) for ZIIP Functions 298
Index 299

Preface

This documentation describes the Natural system commands. It is organized under the following headings:

Issuing System Commands	Describes the general rules that apply when you enter a Natural system command.
System Command Syntax	Explains the symbols that are used within the syntax descriptions of Natural system commands.
System Commands Grouped by Category	Provides an overview of the Natural system commands grouped by category.
System Commands in Alphabetical Order	Descriptions of the system commands in alphabetical order.

1 About this Documentation

- Document Conventions 2
- Online Information and Support 2
- Data Protection 3

Document Conventions

Convention	Description
Bold	Identifies elements on a screen.
`Monospace font`	Identifies service names and locations in the format *`folder.subfolder.service`*, APIs, Java classes, methods, properties.
`Italic`	Identifies: Variables for which you must supply values specific to your own situation or environment. New terms the first time they occur in the text. References to other documentation sources.
`Monospace font`	Identifies: Text you must type in. Messages displayed by the system. Program code.
{ }	Indicates a set of choices from which you must choose one. Type only the information inside the curly braces. Do not type the { } symbols.
|	Separates two mutually exclusive choices in a syntax line. Type one of these choices. Do not type the | symbol.
[ ]	Indicates one or more options. Type only the information inside the square brackets. Do not type the [ ] symbols.
...	Indicates that you can type multiple options of the same type. Type only the information. Do not type the ellipsis (...).

Online Information and Support

Product Documentation

You can find the product documentation on our documentation website at **https://documentation.softwareag.com**.

Product Training

You can find helpful product training material on our Learning Portal at **https://learn.softwareag.com**.

Tech Community

You can collaborate with Software GmbH experts on our Tech Community website at **https://techcommunity.softwareag.com**. From here you can, for example:

- Browse through our vast knowledge base.
- Ask questions and find answers in our discussion forums.
- Get the latest Software GmbH news and announcements.
- Explore our communities.
- Go to our public GitHub and Docker repositories at **https://github.com/softwareag** and **https://hub.docker.com/publishers/softwareag** and discover additional Software GmbH resources.

Product Support

Support for Software GmbH products is provided to licensed customers via our Empower Portal at **https://empower.softwareag.com**. Many services on this portal require that you have an account. If you do not yet have one, you can request it at **https://empower.softwareag.com/register**. Once you have an account, you can, for example:

- Download products, updates and fixes.
- Search the Knowledge Center for technical information and tips.
- Subscribe to early warnings and critical alerts.
- Open and update support incidents.
- Add product feature requests.

Data Protection

Software AG products provide functionality with respect to processing of personal data according to the EU General Data Protection Regulation (GDPR). Where applicable, appropriate steps are documented in the respective administration documentation.

2 Issuing System Commands

- Command Input 6
- Command Line 6
- NEXT Prompt 6
- MORE Prompt 7

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

Notation *vrs* or *vr*

When used in this documentation, the notation *vrs* or *vr* represents the relevant product version (see also *Version* in the *Glossary*).

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

You can enter commands in the command line at the command prompt (===>).

Some system commands may also be available via PF keys or via the main menu.

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 10
- Example of Command Syntax 11

Syntax Elements

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

Element	Explanation
ABCDEF	Upper-case non-italic letters indicate that the term is either a Natural keyword or a Natural reserved word that must be entered exactly as specified.
ABCDEF	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.
ABCDEF	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.
abcdef	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 the 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 the 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 the entire bracketed expression.
Other symbols (except [ ] { } | ... ,... :...)	All other symbols except those defined in this table must be entered exactly as specified. **Exception:** 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 Category

- Navigating in Natural 14
- Natural Development Environment 14
- Managing Applications with Natural Objects 15
- Monitoring, Debugging and Tracing 16
- Natural with Other Products 17

This chapter is a summary of Natural system commands grouped by category.

Navigating in Natural

Command	Brief Description
FIN	Terminates a Natural session.
LAST	Displays the system commands that were last executed, and allows you to execute them again.
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).
MAINMENU	Switches Natural main menu mode on or off, or invokes a program which creates a user-defined menu.
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.

Natural Development Environment

Command	Brief Description
AIV	Displays all user-defined application-independent variables (AIVs) which are currently active.
COMPOPT	Sets compilation options that affect the way in which Natural objects are compiled.
CPINFO	Displays Natural code page settings such as the contents of the system variables *LOCALE and *CODEPAGE, and the code page definitions in the NATCONFG configuration module.
GLOBALS	Changes the settings of Natural session parameters.
HELP	Invokes the Natural help system.
KEY	Assigns functions to keys to be used in your Natural session.
LAST	Displays the system commands that were last executed, and allows you to execute them again.
LIST XREF	Displays all active cross-reference data for the current library. (Only available if Predict is installed.)
NATQVS	Displays information (for example, CPU ID and logical partitions) about the machine on which Natural runs.
SYSAPI	Only applies to products that provide application programming interfaces (APIs). Invokes the SYSAPI utility to locate APIs if provided by the products installed at your site.

Command	Brief Description
SYSCP	Invokes the SYSCP utility to view code page information and administrate code pages for Natural source objects.
SYSEXT	Invokes the SYSEXT utility with Natural application programming interfaces.
SYSEXV	Invokes the SYSEXV utility with examples of the new features of the current Natural versions and debugging hints.
SYSFILE	Invokes the Natural Print/Work Files function of the SYSTP utility for information on the print files and work files available.
SYSPARM	Changes the settings of Natural profile parameters.
SYSPROD	Displays a list of the products installed at your site, and information on these products.
SYSPROF	Displays the current definitions of the Natural system files.
UPDATE	Prevents database updating being carried out by a program.
XREF	Controls the use of the Predict function "active cross-references". This function automatically creates documentation in Predict about the objects which a program/data area references. (Only available if Predict is installed.)

Managing Applications with Natural Objects

Command	Brief Description
CATALL	Catalogs (compiles) *all* objects or selected objects in the current library.
CATALOG	Catalogs (compiles) the source code currently in the editor work area, and if the syntax has been found to be correct, stores the resulting cataloged object in the Natural system file.
CHECK	Checks that the source code currently in the editor work area does not contain any syntax errors. Syntax checking is also performed as part of the system commands RUN, CATALL, CATALOG and STOW.
CLEAR	Clears the contents of the editor work area.
DELETE	Deletes a source object and/or the corresponding cataloged object from the Natural system file.
EDIT	Opens an editor to create or modify source code.
EXECUTE	Executes a cataloged (compiled) program that has been stored as a cataloged object in the Natural system file.
INPL	Invokes the INPL utility. It is *only* used for the loading of installation data sets into the system files.
LIST	Lists one or more objects contained in the current library or the contents of the editor work area.
READ	Transfers a source object from the Natural system file to the editor work area.
RENAME	Changes the name of an object or the name and the type of an object.

Command	Brief Description
RENUMBER	Renumbers the source code currently in the editor work area.
ROUTINES	Shows you which objects in the current library use which external subroutines.
RUN	Compiles and executes the source program currently in the work area of the editor.
SAVE	Stores the source code currently in the editor work area as a source object in the Natural system file.
SCAN	Searches for a character string within a source with the option to replace the string.
STOW	Catalogs (compiles) and stores source code as both a source object and a cataloged object in the current Natural system file.
STRUCT	Performs structural indentation of a program source and helps detecting structural inconsistencies.
SYSDDM	Creates and maintains Natural data definition modules (DDMs).
SYSERR	Creates and maintains the messages you wish your Natural applications to display to the users.
SYSMAIN	Transfers Natural objects within the Natural system from one library to another.
SYSNCP	Creates and maintains the command processors to be used in your Natural applications.
SYSOBJH	Processes Natural and non-Natural objects for distribution in Natural environments.
SYSRPC	Invokes the SYSRPC utility to create and maintain remote procedure calls, that is, provides the settings necessary to execute a subprogram located on a remote server.
UNLOCK	Displays locked Natural objects with the option to unlock them.
WHICH	Lists objects in the libraries used at run time in the Library Search Order.

Monitoring, Debugging and Tracing

Command	Brief Description
BUS	Invokes the Buffer Usage Statistics function of the SYSTP utility. BUS provides statistical information on the usage of Natural buffers.
DUMP	Provides information for support personnel in order to locate an error that caused an abnormal termination (abend) of the Natural system.
HELP	Invokes the Natural help system.
LASTMSG	Displays additional information on the error situation which occurred last.
PROFILER	Invokes the Profiler utility to control and maintain trace data recorded by the Natural Data Collector. See also the system command RDC.
RDC	Used in conjunction with the Profiler utility to activate or deactivate the trace recording function of the Natural Data Collector. See also the system command PROFILER.
RPCERR	Displays the last Natural error number and message if related to Natural RPC (Remote Procedure Call), and the last EntireX Broker reason code and associated message.

Command	Brief Description
SYSADA	Invokes the ADACALL utility to issue Adabas direct calls (native commands) directly to an Adabas database. (Only available if Adabas is installed.)
SYSBPM	Invokes the SYSBPM utility to monitor the buffer pool and adjust it to meet your requirements.
SYSEDT	Invokes the SYSEDT utility to display runtime information of the editor buffer pool, modify its parameters, and delete logical work and recovery files.
SYSTP	Invokes the SYSTP utility to monitor and control TP-monitor-specific characteristics of Natural.
TECH	Displays technical and other information on your Natural session.
TEST	Invokes the Natural debugger for online testing and debugging. The Natural debugger allows you to test various aspects of your applications and locate errors in the processing flow of programs.
TEST DBLOG	Invokes the DBLOG utility for logging database calls.

Natural with Other Products

The following system commands are only available in connection with other products installed at your site:

Product	Command	Brief Description
Adabas	SYSADA	Invokes the ADACALL utility to issue Adabas direct calls (native commands) directly to an Adabas database.
Natural for Db2	LISTSQLB	Invokes the **Explain all SQL Statements** function of **Natural Tools for Db2**. See *Using Natural Tools for Db2* in the *Database Management System Interfaces* documentation.
	SQLDIAG	Provides diagnostic information about the last SQL statement (other than a GET DIAGNOSTICS statement) that was executed. This diagnostic information is gathered as the previous SQL statement is executed. Some of the information available through the GET DIAGNOSTICS statement is also available in the SQLCA.
	SYSDB2	Invokes **Natural Tools for Db2** with functions required to maintain Db2 applications. See *Using Natural Tools for Db2* in the *Database Management System Interfaces* documentation.
	LISTDBRM	Displays existing DBRMs (Natural for Db2) of Natural programs or Natural programs referencing a given DBRM (or package).

Product	Command	Brief Description
	LISTSQL	Generates a list of those Natural statements in the source code of an object which are associated with a database access. Also, it displays the corresponding SQL commands these Natural statements have been translated into. This enables you to view the generated SQLCODE before executing a Natural program which accesses an SQL table.
	SQLERR	Displays diagnostic information about the most recent SQL error.
Natural Optimizer Compiler	NOCOPT	Displays the current settings of the Natural Optimizer Compiler options as they were specified during Natural startup, and allows you to change these settings.
	NOCSHOW	Provides buffer information on the output generated by the PGEN option. PGEN causes the Natural Optimizer Compiler to output generated code and internal Natural structures.
	NOCSTAT	Provides statistical data on programs suitable for processing by the Natural Optimizer Compiler.
Natural Security	MAIL	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.
	PROFILE	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.
Predict	LIST XREF	Displays all active cross-reference data for the current library.
	XREF	Controls the use of the Predict function “active cross-references”. This function automatically creates documentation in Predict about the objects which a program/data area references.
Natural Batch for zIIP, Natural for CICS for zIIP, Natural for Com-plete for zIIP Natural for IMS for zIIP	ZIIP	Invokes statistics on IBM System z Integrated Information Processors (zIIPs) used in a Natural z/OS environment. See also the *Natural for zIIP* documentation.
Various products	SYSAPI	Invokes the SYSAPI utility to locate application programming interfaces (APIs) if provided by the products installed at your site.

5 AIV

`AIV`

This command is used to display all user-defined application-independent variables (AIVs) which are currently active.

On the list displayed, you can mark an AIV with the command `DI` to display the content of the AIV.

You can display the content in alphanumeric or hexadecimal form. To switch from alphanumeric to hexadecimal display and vice versa, you use PF10 and PF11.

For further information, see

- `DEFINE DATA` statement (*Defining Application-Independent Variables*) in the *Statements* documentation;
- *User-Defined Variables* in the *Programming Guide*.

6 BUS

BUS

This command is used to invoke the function Buffer Usage Statistics of the SYSTP utility. It provides information on the buffers allocated for the current Natural session, their sizes, and the actual buffer space being used.

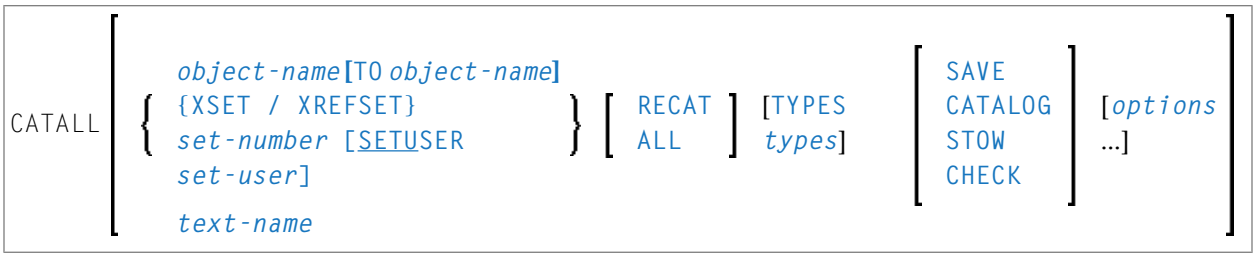
Note: The BUS command performs the same function as the SYSBUS command which is no longer available.

For further information, see *Buffer Usage Statistics* in the *Utilities* documentation.

The information provided by the BUS command can also be obtained with the application programming interface USR1019N. See *SYSEXT - Natural Application Programming Interfaces* in the *Utilities* documentation.

7 CATALL

- Catalog Objects from/to 24
- Predict Set and Set User 25
- Select Cataloged Source Objects Only, or Select All Source Objects 25
- Select Object Types 25
- Select Function 26
- Select Options 26
- Selection List 27
- Direct Command Syntax 28



This command is used to catalog (compile) and store single or multiple Natural objects in the current Natural library.

When you enter the CATALL command without any additional options, the **Catalog Objects in Library** screen is displayed, which you use to perform the functions described below. You can also issue the CATALL command directly using the **command syntax** shown above.

CATALL processes all objects in the order in which the object types are listed on the **Catalog Objects in Library** screen (see also the information for TYPES in the section *Direct Command Syntax*).

You can also select functions on the **Catalog Objects in Library** screen by default using the subprogram CATALLU2. In addition you may enable CATALLU2 to be called in batch or command mode. The subprogram is delivered as a source object in library SYSTEM (FNAT). To activate the subprogram, modify it as described in its source, then catalog it and copy it to SYSLIB. The subprogram is called before the **Catalog Objects in Library** screen is output.

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

Catalog Objects from/to

If you wish CATALL to be performed for *all* objects of the selected types in the current library, specify an asterisk (*) as object name in the **from** field.

If you wish CATALL to be performed for a certain range of objects, you can use asterisk notation (*) and wildcard notation (?) for the name in the **from** field, as described for the system command LIST.

You can also specify a start setting and an end setting for a certain range of objects by entering corresponding object names (without asterisk or wildcard notation) in the **from** and **to** fields.

Instead of entering settings in these fields, you can also select objects from a **selection list**.

Moreover, you can use the **from** field to enter the name of an object of type text which contains a list of CATALL commands. The CATALL commands contained in the text will then be executed. You can either create such a text manually or have it created automatically when you use the **selection list**.

Predict Set and Set User

If you wish CATALL to be performed for the objects of a Predict set, specify in **Predict Set** a one- or two-digit number that identifies the retained set to be used for the names of the objects to be processed. CATALL processes all objects in the order in which they are listed in the Predict set.

If you use a Predict set, the **from** field and the **to** field must not be used. Additionally, you can specify in **Set User** the ID of the user who created the Predict set. If no ID is specified, the value of the system variable *USER is used.

A retained set is created with the save set option of the LIST XREF command.

Note: Predict has to be installed. For detailed information on Predict sets, refer to the *Predict* documentation.

Select Cataloged Source Objects Only, or Select All Source Objects

The following applies:

- If you mark **Select cataloged source objects only**, only the source objects in the current library that have already been cataloged earlier (a corresponding cataloged object exists) are cataloged.
- If you mark **Select all source objects**, *all* source objects (with or without corresponding cataloged objects) in the current library are cataloged.

Select Object Types

By default, CATALL applies to objects of all types in the current library (all object types are marked with X).

If you wish objects of a certain type not to be processed by CATALL, overwrite the respective X with a blank.

Select Function

You can select one of the following functions to be applied to the selected objects: SAVE, CATALOG, STOW or CHECK. The functions correspond to the system commands of the same names.

Note: Objects of type copycode and text will be saved, even if you select Stow. They will not be saved, if you select Catalog.

Select Options

You can select one or more of the following options for CATALL processing:

Condition Code in Batch	If you execute CATALL in batch mode and mark this option with a character, Condition Code 55 will be returned either if a syntax error is detected during CATALL execution or if no objects are found within the specified range of objects to be processed (applies to CATALOG and STOW only).
Renumber Source-Code Lines	By default, the source-code lines of sources that were saved or stowed are also renumbered. If you wish no automatic renumbering of lines, overwrite the X in this field with a blank.
Keep Result List	CATALL generates a result list. If you wish to keep this list for further use, mark this field with a character. The library SYSEXT contains the application programming interface USR1024N, which you can use to output the result list. You can also re-display the result list with another CATALL command. Since the parameters are also stored in the result list, the parameters of the CATALL which created the result list are valid. In batch, if a library contains a result list, it is displayed automatically with a CATALL command. In this case, the batch CATALL job will issue a message and because no objects will be cataloged the job will end with Condition Code 56. Online, if a library contains a result list, you will be asked whether to display the previous result list or start a new CATALL run.
Processing Information	During online processing, CATALL shows a scrolling display of processing status information. During batch processing, CATALL only outputs the objects which caused an error. To suppress this display, overwrite the X in this field with a blank.
Error Report	At the end of processing, CATALL displays a list of the errors that occurred. To suppress this error list, overwrite the X in this field with a blank.

Extended Error Report	The error report will be output in extended form, with directory information, error line and error message. To output the Extended Error Report, mark this field with an X.
PF4 AddOp	Pressing PF4 causes a window to appear in which you can select or enter additional options. Report Text Object: Enter the name of a Natural text object. An error report of a CATALL run will be written to this text object.

Selection List

If you wish to use CATALL only for certain objects, you can select these objects from a selection list.

To do so, first make the desired specifications under Select Function and Select Options, and then press PF5. A list of the objects stored in the current library is displayed.

The list corresponds to that of the system command LIST. Scrolling the selection list and the specification of new selection criteria on the list is also done in the same way as with the LIST command.

On the list, select the desired objects by marking them with a character in the column Cmd. To simultaneously select *all* objects of the current selection list, press PF5. You can then scroll the list, specify other selection criteria, and select further objects.

When you have selected all objects you wish to process, press PF3.

A window is displayed which allows you to store the selected set of objects so that you can re-use it in other CATALL processing:

- If you enter a name in the window, the selected set of objects will automatically be stored (in the form of CATALL commands) in an object of type text of that name. You can later use that text name in the field **Catalog Objects from** of the **Catalog Objects in Library** screen.
- If this is not desired, press ENTER without entering anything in the window.

CATALL will then begin to process the selected objects.

Direct Command Syntax

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

object-name TO *object-name*	Corresponds to the fields **Catalog Objects from** and **to** of the **Catalog Objects in Library** screen, see ***Catalog Objects from/to***.
{XSET / XREFSET} *set-number*	Corresponds to the field **Predict Set** of the **Catalog Objects in Library** screen, see ***Predict Set and Set User***.
[SETUSER *set-user*]	Corresponds to the field **Set user** of the **Catalog Objects in Library** screen, see ***Predict Set and Set User***.
RECAT / ALL	Corresponds to the options **Select cataloged source objects only** (RECAT) and **Select all source objects** (ALL) of the **Catalog Objects in Library** screen. RECAT is the default. See also ***Select Cataloged Source Objects Only, or Select All Source Objects***.
TYPES *types*	Corresponds to the object types listed on the **Catalog Objects in Library** screen. Possible *types* (processed in the order below) are: G Global data areas L Local data areas A Parameter data areas C Copycodes T Texts 7 Functions N Subprograms S External subroutines H Helproutines M Maps 8 Adapters P Programs 4 Classes * All types (this is the default) The *types* have to be specified as *one* 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.
SAVE / CATALOG / STOW / CHECK	Corresponds to the actions of the same names on the **Catalog Objects in Library** screen, see ***Select Function***. CATALOG is the default.

options	These options correspond to **Select Options** on the **Catalog Objects in Library** screen, see ***Select Options***. Possible *options* are:	
	CC	Condition Code will be returned.
	NOREN	No automatic renumbering of source-code lines.
	KEEP	Result list will be kept.
	NOSCROLL	Online: scrolling display of processing status information will be suppressed. Batch: output of only those objects, which caused an error.
	NOREPORT	Error report will be suppressed.
	FULL	Error report will be extended.
	EL *text-object* [R]	
	EL *text-object*	Output error list to a Natural text object.
	R	If an object already exits, the EL parameter is disabled, unless R (replace) is specified behind *text-object*.
	Note: If you specify NOREPORT *and* NOSCROLL, KEEP will automatically apply, too.	
text-name	Corresponds to specifying a text name in the **Catalog Objects from** field of the **Catalog Objects in Library** screen, see ***Catalog Objects from/to***.	

Examples:

To stow only source objects for which a cataloged object already exist

- Enter the following command:

```
CATALL * STOW KEEP CC NOREN ↩
```

The above command is with implicit RECAT and has the same effect as the following command.

```
CATALL * RECAT STOW KEEP CC NOREN ↩
```

To stow all objects

- Enter the following command:

```
CATALL * ALL STOW KEEP CC NOREN
```

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).

8 CATALOG

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

Related commands: SAVE | STOW.

This command is used to catalog (compile) the source code currently in the work area of a Natural editor and (if the syntax has been found to be correct) store the resulting cataloged object in the current Natural system file.

See also:

Natural Compiler in Natural *System Architecture*
Object Naming Conventions in *Using Natural*

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.

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

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

9 CHECK

CHECK

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

Mode	Reaction
Online	Syntax checking stops when a syntax error is detected, and the source-code line that contains the error is marked with E in the corresponding editor.
Batch	Syntax checking continues until all statements have been checked, and all errors detected are included in the output listing.

Note: Syntax checking is also performed as part of the RUN, STOW, CATALOG and CATALL commands.

See also *Natural Compiler* in Natural *System Architecture*.

10 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.

11 COMPOPT

- COMPOPT Syntax Explanation 38
- Specifying Compiler Keyword Parameters 38
- General Compilation Options 39

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

This system command is used to set various compilation options. The options are evaluated when a Natural 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 keyword subparameters of the parameter macro NTCMPO in the Natural parameter module or in the profile parameter CMPO. When you change the library, the COMPOPT options are reset to their default values.

COMPOPT Syntax Explanation

COMPOPT	If you issue the COMPOPT system command without options, the **Compilations Options** screen appears. The keywords available there are described below.
COMPOPT *option=value*	The keywords for the individual options are described below. The setting assigned to a compiler option is in effect until you issue the next LOGON command to another library. At LOGON, the default settings set with the macro NTCMPO and/or profile parameter CMPO will be resumed.

Specifying Compiler Keyword Parameters

You can specify compiler keyword parameters on different levels:

1. The default settings of the individual keyword parameters are specified in the macro NTCMPO in the Natural parameter module.
2. At session start, you can override the compiler keyword parameters with the profile parameter CMPO.
3. During an active Natural session, there are two ways to change the compiler keyword parameters with the COMPOPT system command: either directly using command assignment (COMPOPT *option=value*) or by issuing the COMPOPT command without keyword parameters 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, the default settings set with the macro NTCMPO and/or the profile parameter CMPO (see 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 object (for example: program, subprogram), you can set compiler parameters (options) 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 object, but do not update settings set with the command `COMPOPT`.

General Compilation Options

The compilation options correspond to the keyword subparameters of the `CMPO` profile parameter and/or the `NTCMPO` parameter macro.

- CHKRULE - Validate INCDIR Statements in Maps
- CPAGE - Code Page Support for Alphanumeric Constants
- DBSHORT - Interpretation of Database Short Field Names
- DB2ARRY - Support Db2 Arrays in SQL SELECT and INSERT Statements
- DB2BIN – Generate SQL Binary Data Types for Natural Binary Fields
- DB2PKYU – Place Primary Key Fields into the Natural DML UPDATE Statement
- DB2TSTI – Generate SQL TIMESTAMP Data Type for Natural TIME Fields
- ECHECK - Existence Check for Object Calling Statements
- GDASC - GDA Signature Check
- GFID - Generation of Global Format IDs
- KCHECK - Keyword Checking
- LOWSRCE - Allow Lower-Case Source
- MASKCME - MASK Compatible with MOVE EDITED
- MAXPREC – Maximum Number of Digits after Decimal Point
- MEMOPT - Memory Optimization for Locally Declared Variables
- NMOVE22 - Assignment of Numeric Variables of Same Length and Precision
- PCHECK - Parameter Check for Object Calling Statements
- PSIGNF - Internal Representation of Positive Sign of Packed Numbers
- THSEP - Dynamic Thousands Separator
- TQMARK - Translate Quotation Mark
- TSENABL - Applicability of TS Profile Parameter

- V82COMP - Disallow Syntax Not Available on Natural for Mainframe 8.2

CHKRULE - Validate INCDIR Statements in Maps

The CHKRULE option can be used to enable or disable a validation check during the catalog process for maps.

ON	INCDIR validation is enabled. If the file (DDM) or field referenced in the INCDIR control statement does not exist, syntax error NAT0721 is raised at compile time. When a Natural map is created, you may include fields which are already defined inside another existing object. This works with nearly all kinds of objects which allow you to define variables and also with DDMs. When the included field is a database variable, it is a map editor built-in behavior to automatically add (besides the included field) an additional INCDIR statement in the map statement body to trigger a Predict rule upload and incorporation when the map is compiled (STOW). The function is similar to what is happening when an INCLUDE statement is processed. However, instead of getting the source lines from a copycode object, they are received from Predict. The search key to find the rule(s) are the DDM name (which is regarded as the file name) and the field name. Both are indicated in the INCDIR statement. An INCDIR rule requested at compile time has not got to be found on Predict, as there is absolutely no requirement for its existence. That implies, it is by no means an error situation if a searched rule is not found. When fields are incorporated from a DDM into a map, the corresponding INCDIR statements are created, including the current DDM and field name as "search key" to request existent rules from Predict. However, if the DDM is renamed after the copy process, the old DDM name (which is not valid anymore) still continues to be used in the INCDIR statement. This causes that no rule is loaded and the programmer is not informed about this. Moreover, it is not only a DDM rename causing this situation. The more likely situation effecting this consequence is to have a wrong FDIC file assigned, by any mistake. In this case, the DDM name is valid, but it cannot be found on the current Predict system file. Then the result is same as when the DDM does not exist at all; the processing rules supposed to be added from Predict are not included.
OFF	INCDIR validation is disabled. This is the default value.

CPAGE - Code Page Support for Alphanumeric Constants

The CPAGE option can be used to activate a conversion routine which translates all alphanumeric constants (from the code page that was active at compilation time into the code page that is active at runtime) when the object is started at runtime.

See also *CPAGE Compiler Option* in the *Unicode and Code Page Support* documentation.

ON	Code page support for alpha strings is enabled.
OFF	Code page support for alpha strings is disabled. This is the default value.

DBSHORT - Interpretation of Database Short Field Names

A database field defined in a DDM is described by two names:

- the short name with a length of 2 characters, used by Natural to communicate with the database (especially with Adabas);
- the long name with a length of 3-32 characters (1-32 characters, if the underlying database type accessed is Db2/SQL), which is supposed to be used to reference the field in the Natural programming code.

Under special conditions, you may reference a database field in a Natural program with its short name instead of the long name. This applies if running in Reporting Mode without Natural Security and if the database access statement contains a reference to a DDM instead of a view.

The decision if a field name is regarded as a short-name reference depends on the name length. When the field identifier consists of two characters, a short-name reference is assumed; a field name with another length is considered as a long-name reference. This standard interpretation rule for database fields can additionally be influenced and controlled by setting the compiler option DBSHORT to ON or OFF:

ON	The usage of a short name is allowed for referencing a database field. However, a data base short name is *not permitted* in general (even if DBSHORT=ON) ■ for the definition of a field when a view is created; ■ when a DEFINE DATA LOCAL statement was specified; ■ when running under Natural Security. This is the default value.
OFF	A database field may only be referenced via its long name. Every database field identifier is considered as a long-name reference, regardless of its length. If a two character name is supplied which can only be found as a short name but not as a long name, syntax error NAT0981 is raised at compile time. This makes it possible to use long names defined in a DDM with 2-byte identifier length. This option is essential if the underlying database you access with this DDM is SQL (Db2) and table columns with a two character name exist. For all other database types (for example, Adabas), however, any attempt to define a long-field with a 2-byte name length will be rejected at DDM generation. Moreover, if no short-name references are used (what can be enforced via DBSHORT=OFF), the program becomes independent of whether it is compiled under Natural Security or not.

Examples:

Assume the following data base field definition in the DDM EMPLOYEES:

Short Name	Long Name
AA	PERSONNEL-ID

Example 1:

```
OPTIONS DBSHORT=ON
READ EMPLOYEES
  DISPLAY AA      /* data base short name AA is allowed
END
```

Example 2:

```
OPTIONS DBSHORT=OFF
READ EMPLOYEES
  DISPLAY AA      /* syntax error NAT0981, because DBSHORT=OFF
END
```

Example 3:

```
OPTIONS DBSHORT=ON
DEFINE DATA LOCAL
1 V1 VIEW OF EMPLOYEES
  2  PERSONNEL-ID
END-DEFINE
READ V1 BY PERSONNEL-ID
  DISPLAY AA     /* syntax error NAT0981, because PERSONNEL-ID is defined in view;
                 /* (even if DBSHORT=ON)
END-READ
END
```

DB2ARRY - Support Db2 Arrays in SQL SELECT and INSERT Statements

The DB2ARRY option can be used to activate retrieval and/or insertion of multiple rows from/into Db2 by a single SQL SELECT or INSERT statement execution. This allows the specification of arrays as receiving fields in the SQL SELECT and as source fields in the SQL INSERT statement. If DB2ARRY is ON, it is no longer possible to use Natural alphanumeric arrays for Db2 VARCHAR/GRAPHIC columns. Instead of these, long alphanumeric Natural variables have to be used.

ON	Db2 array support is enabled.
OFF	Db2 array support is not enabled. This is the default value.

DB2BIN – Generate SQL Binary Data Types for Natural Binary Fields

The DB2BIN option can be used to support the Db2 data types BINARY and VARBINARY.

If DB2BIN is set to OFF, Natural binary fields (format B(*n*)) are generated as SQL data type CHAR (*n*<= 253) or VARCHAR (253<*n*<=32767) like it was in previous releases. DB2BIN=OFF is good for those who used Natural binary fields like SQL CHAR fields. B2 and B4 are treated as SQL SMALLINT and INTEGER.

If DB2BIN is set to ON, Natural binary fields (format B(*n*)) are generated as SQL data type BINARY (*n*<=255) or VARBIN (255<*n*<=32767). DB2BIN=ON is good for those who want to use SQL binary columns. B2 and B4 are also treated as SQL BINARY(2) and BINARY(4).



Note: The setting of DB2BIN at the end of the compilation is used for the complete Natural object. It cannot be changed for parts of a Natural object.

ON	SQL types BINARY and VARBIN are generated for Natural binary fields.
OFF	SQL types CHAR and VARCHAR are generated for Natural binary fields, except B2 and B4. The latter are treated as SQL data types SMALLINT and INTEGER. This is the default value.

DB2PKYU – Place Primary Key Fields into the Natural DML UPDATE Statement

Only applies if supported by the Natural for Db2 version installed at your site.

The DB2PKYU option can be used to update Db2 primary key fields with a Natural DML UPDATE statement. Db2 primary key fields are fields whose short names begin with the character 0 in the DDM.



Note: The setting of DB2PKYU at the end of the compilation is used for the complete Natural object. It cannot be changed for parts of a Natural object.

ON	Db2 primary key fields are updated. Db2 primary key fields which are updated within the Natural program are placed into the resulting Db2 positioned UPDATE statement of a Natural DML UPDATE statememt. The SQLCODE +535 Db2 returned for this positioned UPDATE is treated as 0 (zero) by the Natural for Db2 runtime system.
OFF	Db2 primary key fields are not updated. Db2 primary key fields which are updated within the Natural program are not placed into the resulting Db2 positioned UPDATE statement.

	This is the default value.

DB2TSTI – Generate SQL TIMESTAMP Data Type for Natural TIME Fields

This option is used to map Natural `TIME` variables to the SQL TIMESTAMP data type instead of the default SQL TIME data type.

`ON`	SQL type TIMESTAMP is generated for Natural `TIME` fields of Natural data format T. This applies to the entire Natural object. You cannot generate only part of an object with the `DB2TSTI` setting.
`OFF`	SQL type TIME is generated for Natural `TIME` fields of Natural data format T. This is the default value.



Note: A Natural `TIME` field only contains tenth of seconds as precision while a SQL TIMESTAMP column can contain a much greater precision. Thus, the TIMESTAMP value read from the SQL database may be truncated if `DB2TSTI=ON` is set.

ECHECK - Existence Check for Object Calling Statements

`ON`	The compiler checks for the existence of an object that is specified in an object calling statement, such as `FETCH [RETURN/REPEAT]`, `RUN [REPEAT]`, `CALLNAT`, `PERFORM`, `INPUT USING MAP`, `PROCESS PAGE USING`, function call, and helproutine call. The existence check is based on a search for the cataloged object or for the source of the object when it is invoked by a `RUN [REPEAT]` statement. It requires that the name of the object to be called/run is defined as an alphanumeric constant (not as an alphanumeric variable). Otherwise, `ECHECK=ON` will have no effect. **Error Control for ECHECK=ON** The existence check is executed only when the object does not contain any syntax errors. The existence check is executed for every object calling statement. The existence check is controlled by the `PECK` profile parameter (see the *Parameter Reference* documentation). **Problems in Using the CATALL Command with ECHECK=ON** When a `CATALL` system command is used in conjunction with `ECHECK=ON`, you should consider the following: If a `CATALL` process is invoked, the order in which the objects are compiled depends primarily on the type of the object and secondarily on the alphabetical name of the object. The object type sequence used is:

	GDAs, LDAs, PDAs, functions, subprograms, external subroutines, help routines, maps, adapters, programs, classes. Within objects of the same type, the alphabetical order of the name determines the sequence in which they are cataloged. As mentioned above, the success of the object calling statement is checked against the compiled form of the called object. If the calling object (the one which is being compiled and includes the object calling statement) is cataloged before the invoked object, the ECHECK result may be wrong if the called object was not cataloged beforehand. In this case, the object image of the called object has not yet been produced by the CATALL command. Solution: ■ Set compiler option ECHECK to OFF. ■ Perform a general compile with CATALL on the complete library, or if just one or a few objects were changed, perform a separate compile on these objects. ■ Set compiler option ECHECK=ON. ■ On the complete library, perform a general compile with CATALL, selecting function CHECK.
OFF	No existence check is performed. This is the default setting.

GDASC - GDA Signature Check

This option is used to store the structural information of a GDA (global data area). This GDA information is called a "GDA signature". A GDA signature is a hash value calculated from the GDA data definitions including the variable name, format, length, and initial values.

- Structural Information of the GDA
- How to Find out the Current Settings of the GDA Signatures
- Benefit of Using GDA Signatures
- Enabling GDA Signatures

Structural Information of the GDA

ON	When a GDA is cataloged, the GDA signature is stored in the cataloged GDA.
OFF	When a GDA is cataloged, the GDA is stored without a GDA signature. This is the default value.

When a Natural object that references a GDA is cataloged and the GDA contains a GDA signature, that GDA signature is copied into the cataloged object, regardless of the current GDASC setting.

At execution time, the GDA signature of the executed Natural object is compared to the GDA signature of the referenced GDA. If both signatures exist and are identical, the GDA is accepted, even if the timestamps are not identical.

The following table describes in detail what is compared at execution time and the expected result.

GDA Contains Signature	Natural Object Contains Signature	Compare	Result
yes	yes	Signatures	Error if not equal
yes	no	Timestamps	Error
no	yes	Timestamps	Error
no	no	Timestamps	Error if not equal

How to Find out the Current Settings of the GDA Signatures

The **LIST DIRECTORY** command shows the field `GDA signature` when a GDA or a Natural object contains a GDA signature.

For a GDA, the GDA signature will be checked for its existence. For a Natural object, the GDA signature will be checked against the signature of the used GDA.

See the following table with the settings of the field `GDA signature`:

GDA Contains Signature	Natural Object Contains Signature	GDA Signature in LIST DIRECTORY of a GDA	GDA Signature in LIST DIRECTORY of a Natural Object
yes	yes	Generated	Identical / Not identical
yes	no	Generated	Only in GDA
no	yes	-*1	Only in object
no	no	-*1	-*2



Notes:

1. `*1`: Show GDA timestamp when GDA was cataloged.
2. `*2`: Show GDA timestamps when the GDA and the Natural object were cataloged.

Benefit of Using GDA Signatures

The GDA signature of a GDA persists if it is merely recataloged but not modified, or if only the comments are modified within the GDA. In these cases, Natural objects referencing the GDA still retain the same GDA signature as the GDA and therefore do not need to be recataloged.

Enabling GDA Signatures

To enable GDA signatures, set `GDASC` to `ON`.

When you catalog a GDA for the first time with `GDASC` set to `ON`, you must also catalog all referencing Natural objects. GDA signatures will then be used at execution time.

Other libraries that do not (yet) have the GDA cataloged with `GDASC` set to `ON` will continue to use GDA timestamps for GDA validation checks. The Natural objects remain without GDA signature.

GFID - Generation of Global Format IDs

This option allows you to control Natural's internal generation of global format IDs so as to influence Adabas's performance concerning the re-usability of format buffer translations.

`ON`	Global format IDs are generated for all views. This is the default value.
`VID`	Global format IDs are generated only for views in local/global data areas, but not for views defined within programs.
`OFF`	No global format IDs are generated.

For details on global format IDs, see the Adabas documentation.

Rules for Generating GLOBAL FORMAT-IDs in Natural

- **For Natural nucleus internal system-file calls:**

```
GFID=abccddee
```

where	equals
a	x'F9'
b	x'22' or x'21' depending on DB statement
cc	physical database number (2 bytes)
dd	physical file number (2 bytes)
ee	number created by runtime (2 bytes)

- **For user programs or Natural utilities:**

```
GFID=abbbbbb
```

where	equals
a	x'F8' or x'F7' or x'F6' where: F6=UPDATE SAME F7=HISTOGRAM F8=all others
bbbbbbb	bytes 1-7 of STOD value

Note: STOD is the return value of the store clock machine instruction (STCK).

KCHECK - Keyword Checking

ON	Field declarations in an object will be checked against a set of critical Natural keywords. If a variable name defined matches one of these keywords, a syntax error is reported when the object is checked or cataloged.
OFF	No keyword check is performed. This is the default value.

The section *Performing a Keyword Check* (in the *Programming Guide*) contains a list of the keywords that are checked by the KCHECK option.

The section *Alphabetical List of Natural Reserved Keywords* (in the *Programming Guide*) contains an overview of all Natural keywords and reserved words.

LOWSRCE - Allow Lower-Case Source

This option supports the use of lower or mixed-case program sources on z/OS platforms. It facilitates the transfer of programs written in mixed/lower-case characters from other platforms to a z/OS environment.

ON	Allows any kind of lower/upper-case characters in the program source.
OFF	Allows upper-case mode only. This requires keywords, variable names and identifiers to be defined in upper case. This is the default value.

When you use lower-case characters with LOWSRCE=ON, consider the following:

- The syntax rules for variable names allow lower-case characters in subsequent positions. Therefore, you can define two variables, one written with lower-case characters and the other with upper-case characters.

  Example:

```
DEFINE DATA LOCAL
1 #Vari  (A20)
1 #VARI  (A20)
```

With LOWSRCE=OFF, these variables are treated as different variables.

With LOWSRCE=ON, the compiler is *not* case sensitive and does not make a distinction between lower/upper-case characters. This will lead to a syntax error because a duplicate definition of a variable is not allowed.

- Using the session parameter EM (Edit Mask) in an I/O statement or in a MOVE EDITED statement, there are characters which influence the layout of the data setting assigned to a variable (EM control characters), and characters which insert text fragments into the data setting.

  Example:

```
#VARI :='1234567890'
  WRITE #VARI (EM=XXXXXxxXXXXX)
```

  With LOWSRCE=OFF, the output is "12345xx67890", because for alpha-format variables only upper-case X, H and circumflex accent (^) sign can be used.

  With LOWSRCE=ON, the output is "1234567890", because an x character is treated like an upper-case X and, therefore, interpreted as an EM control character for that field format. To avoid this problem, enclose constant text fragments in apostrophes (').

  Example:

```
WRITE #VARI(EM=XXXXX'xx'XXXXX)
```

  The text fragment is *not* considered an EM control character, regardless of the LOWSRCE settings.

- Since all variable names are converted to upper-case characters with LOWSRCE=ON, the display of variable names in I/O statements (INPUT, WRITE or DISPLAY) differs.

  Example:

```
MOVE 'ABC' to #Vari
  DISPLAY #Vari
```

  With LOWSRCE=OFF, the output is:

```
     #Vari
--------------------

ABC
```

With LOWSRCE=ON, the output is:

```
     #VARI
--------------------

ABC
```

MASKCME - MASK Compatible with MOVE EDITED

ON	The range of valid year values that match the YYYY mask characters is 1582 - 2699 to make the MASK option compatible with MOVE EDITED. If the profile parameter MAXYEAR is set to 9999, the range of valid year values is 1582 - 9999.
OFF	The range of valid year values that match the YYYY mask characters is 0000 - 2699. This is the default value. If the profile parameter MAXYEAR is set to 9999, the range of valid year values is 0000 - 9999.

MAXPREC – Maximum Number of Digits after Decimal Point

This option determines the maximum number of digits after the decimal point that the Natural compiler generates for results of arithmetic operations.

7, ..., 29	The value denotes the maximum number of digits after the decimal point that the Natural compiler generates for results of arithmetic operations. The default value 7 provides upwards compatibility for existing applications. If such applications are cataloged with MAXPREC=7, they will deliver the same results as before. Objects cataloged with a Natural version that did not support the MAXPREC option are executed as if MAXPREC=7 had been set. If higher precision is desired for intermediate results, the value should be increased. The setting of MAXPREC does not limit the number of digits after the decimal point that can be specified for user defined fields and constants. However, the precision of such fields and constants influences the precision of results of arithmetic operations. This makes it possible to benefit from enhanced precision in selected computations without having the need to set the compiler option MAXPREC to a value that unintentionally affects other computations. So even if MAXPREC=7 is in effect, the following example program can be cataloged and executed:

	``` DEFINE DATA LOCAL 1 P (P1.15) END-DEFINE P := P + 0.1234567890123456 END ``` See also *Precision of Results of Arithmetic Operations* in the *Programming Guide*.

Caution: Changing the value of the MAXPREC option that is being used to catalog a Natural object may lead to different results, even if the object source has not been changed. See example below.

Example:

```
DEFINE DATA LOCAL
1 #R (P1.7)
END-DEFINE
#R := 1.0008 * 1.0008 * 1.0008
IF #R = 1.0024018 THEN ... ELSE ... END-IF
```

The value of #R after the computation and the execution of the IF statement depend on the setting of MAXPREC:

Setting of MAXPREC Effective at Compile Time	Value of #R	Executed Clause of IF Statement
MAXPREC=7	1.0024018	THEN clause
MAXPREC=12	1.0024019	ELSE clause

MEMOPT - Memory Optimization for Locally Declared Variables

This option determines whether or not memory is allocated for unused level-1 fields or groups defined locally (DEFINE DATA LOCAL).

ON	Storage is allocated only for ■ level-1 field, if the field or a redefinition thereof is accessed; ■ group, if the group or at least a group-field is accessed.
OFF	Data storage is allocated for all groups and fields declared locally. This is the default setting.

NMOVE22 - Assignment of Numeric Variables of Same Length and Precision

ON	Assignments of numeric variables where source and target have the same length and precision is performed as with Natural Version 2.2.
OFF	Assignments of numeric variables where source and target have the same length and precision is performed as with Natural Version 2.3 and above, that is they are processed as if source and target would have different length or precision. This is the default value.

PCHECK - Parameter Check for Object Calling Statements

ON	The compiler checks the number, format, length and array index bounds of the parameters that are specified in an object calling statement, such as CALLNAT, PERFORM, INPUT USING MAP, PROCESS PAGE USING, function call, and helproutine call. Also, the OPTIONAL feature of the DEFINE DATA PARAMETER statement is considered in the parameter check. The parameter check is based on a comparison of the parameters of the object calling statement with the DEFINE DATA PARAMETER definitions for the object to be invoked. It requires that ■ the name of the object to be called is defined as an alphanumeric constant (not as an alphanumeric variable), ■ the object to be called is available as a cataloged object. Otherwise, PCHECK=ON will have no effect. **Error Control for PCHECK=ON** The parameter check is executed only when the object does not contain any syntax errors. The parameter check is executed for every object calling statement. The parameter check is controlled by the PECK profile parameter (see the *Parameter Reference* documentation). **Problems in Using the CATALL Command with PCHECK=ON** When a CATALL command is used in conjunction with PCHECK=ON, you should consider the following: If a CATALL process is invoked, the order in which the objects are compiled depends primarily on the type of the object and secondarily on the alphabetical name of the object. The object type sequence used is: GDAs, LDAs, PDAs, functions, subprograms, external subroutines, help routines, maps, adapters, programs, classes. Within objects of the same type, the alphabetical order of the name determines the sequence in which they are cataloged. As mentioned above, the parameters of the object calling statement are checked against the compiled form of the called object. If the calling object (the one which is being compiled and includes the object

	calling statement) is cataloged before the invoked object, the PCHECK result may be wrong if the parameters in the invoking statement and in the called object were changed. In this case, the new object image of the called object has not yet been produced by the CATALL command. This causes the *new* parameter layout in the object calling statement to be compared with the *old* parameter layout of the DEFINE DATA PARAMETER statement of the called subprogram. Solution: ■ Set compiler option PCHECK to OFF. ■ Perform a general compile with CATALL on the complete library, or if just one or a few objects were changed, perform a separate compile on these objects. ■ Set compiler option PCHECK=ON. ■ On the complete library, perform a general compile with CATALL, selecting function CHECK.
OFF	No parameter check is performed. This is the default setting.

PSIGNF - Internal Representation of Positive Sign of Packed Numbers

ON	The positive sign of a packed number is represented internally as H'F'. This is the default value.
OFF	The positive sign of a packed number is represented internally as H'C'.

THSEP - Dynamic Thousands Separator

This option can be used to enable or disable the use of thousands separators at compilation time. See also the profile and session parameter THSEPCH and the section *Customizing Separator Character Displays* (in the *Programming Guide*).

ON	Thousands separator used. Every thousands separator character that is not part of a string literal is replaced internally with a control character.
OFF	Thousands separator not used, i.e. no thousands separator control character is generated by the compiler. This is the compatibility setting.

TQMARK - Translate Quotation Mark

ON	Each double quotation mark within a text constant is output as a single apostrophe. This is the default value.
OFF	Double quotation marks within a text constant are not translated; they are output as double quotation marks.

Example:

```
RESET A(A5)
A:= 'AB"CD'
WRITE '12"34' / A / A (EM=H(5))
END
```

With TQMARK ON, the output is:

```
12'34
AB'CD
C1C27DC3C4
```

With TQMARK OFF, the output is:

```
12"34
AB"CD
C1C27FC3C4
```

TSENABL - Applicability of TS Profile Parameter

This option determines whether the profile parameter TS (translate output for locations with non-standard lower-case usage) is to apply only to Natural system libraries (that is, libraries whose names begin with "SYS", except SYSTEM) or to all user libraries as well.

Natural objects cataloged with TSENABL=ON determine the TS parameter even if they are located in a non-system library.

ON	The profile parameter TS applies to all libraries.
OFF	The profile parameter TS only applies to Natural system libraries. This is the default value.

V82COMP - Disallow Syntax Not Available on Natural for Mainframe 8.2

The V82COMP option checks whether the syntax of the features provided since Natural for z/OS Version 9.2 is also supported by Natural for Mainframes Version 8.2. If any syntax incompatibilities between these two versions are detected, the compilation fails with an appropriate Natural error message and reason code.

The following values are possible:

ON	When a program is compiled, every attempt to use a syntax construction that is supported by Natural for z/OS 9.2, but not in version 8.2, is rejected with a NAT0597 syntax error and an appropriate **reason code** (see the following section).
OFF	No syntax check is performed. This is the default value.

Reason Codes for Syntax Errors

The following reason codes indicate which syntax parts are not supported by Natural for Mainframe 8.2:

Reason Code	Invalid Syntax on Natural for Mainframe 8.2
019	The following new system variable was used: `*TIMESTMPX`

020	The `COMPRESS` statement allows to use a field of type `LOGICAL`.
	
	Example:
	
	`DEFINE DATA LOCAL`
	`1 #L (L)  INIT <TRUE>`
	`1 #RESULT (A50)`
	`END-DEFINE`
	`COMPRESS 'The value of #L is' #L INTO #RESULT`

021	The `COMPRESS` statement allows to provide an edit mask for each field.
	
	Example:
	
	`DEFINE DATA LOCAL`
	`1 #A (A10)  INIT <'ABCDEF'>`
	`1 #P (P5.3)  INIT <-12.345>`
	`1 #L (L)  INIT <TRUE>`
	`1 #RESULT (A50)`
	`END-DEFINE`
	`COMPRESS #A (EM=X_X)  #P(EM=-ZZZ9.9)  #L(EM=FALSE/TRUE) INTO #RESULT`

022	The work file number can be provided with a `CONST` variable in a `DEFINE WORK`, `WRITE WORK`, `READ WORK`, and `CLOSE WORK` statement.
	
	Example:
	
	`DEFINE DATA LOCAL`
	`1 #WNBR (I2)  CONST <12>`
	`END-DEFINE`
	`WRITE WORK FILE #WNBR 'ABC'`
	`CLOSE WORK FILE #WNBR`

023	The `READ .. BY ISN` statement was enhanced to allow backward reading in DESCENDING order. However, the usage of this feature requires to run against Adabas version 8.6 (or higher). Example:

Reason Code	Invalid Syntax on Natural for Mainframe 8.2

	`DEFINE DATA LOCAL`
	`1 VIEW1 VIEW OF EMPLOYEES`
	`2 NAME`
	`END-DEFINE`
	`READ (10) VIEW1 DESCENDING BY ISN = 1000    /* Reads 10 records starting ↩ with ISN=1000`
	`DISPLAY *ISN NAME                          /* (or the next lower) in ↩ descending order`
	`END-READ`

024	The READ .. BY ISN statement was enhanced to allow an end value sent to the database by using the option TO between start and end value. The usage of this feature requires to run against Adabas version 8.6 (or higher).
	
	Example:
	
	`DEFINE DATA LOCAL`
	`1 VIEW1 VIEW OF EMPLOYEES`
	`2 NAME`
	`END-DEFINE`
	`READ VIEW1 BY ISN = 11  TO  20    /* Reads all records between ISN=11 and ↩ 20.`
	`DISPLAY *ISN NAME                          /* At loop end, the record in ↩ VIEW1 is the last record found in this range.`
	`END-READ`

025	The END-XXX statement was enhanced to allow the use of label name or line number.
	
	Example 1:
	
	`DEFINE DATA LOCAL`
	`1 EMPLOY-VIEW VIEW OF EMPLOYEES`
	`  2 PERSONNEL-ID`
	`  2 NAME`
	`  2 JOB-TITLE`
	`  2 CITY`
	`END-DEFINE`
	`FD. FIND EMPLOY-VIEW WITH CITY = 'PARIS'`
	`         WHERE JOB-TITLE = 'INGENIEUR COMMERCIAL'`
	`         DISPLAY NOTITLE`
	`         CITY JOB-TITLE PERSONNEL-ID NAME`
	`END-FIND (FD.)`
	`END`
	
	Example 2:

Reason Code	Invalid Syntax on Natural for Mainframe 8.2
	0010 DEFINE DATA LOCAL 0020 1 EMPLOY-VIEW VIEW OF EMPLOYEES 0030 2 PERSONNEL-ID 0040 2 NAME 0050 2 JOB-TITLE 0060 2 CITY 0070 END-DEFINE 0080 FD. FIND EMPLOY-VIEW WITH CITY = 'PARIS' 0090 WHERE JOB-TITLE = 'INGENIEUR COMMERCIAL' 0100 DISPLAY NOTITLE 0110 CITY JOB-TITLE PERSONNEL-ID NAME 0120 END-FIND (0080) 0130 END

12 CPINFO

This command is used to display all relevant Natural code page settings, such as content of the system variables *LOCALE, *CODEPAGE, current code page of the source area, current settings of the relevant parameters, ICU version, Unicode version, etc, and to display the code pages defined in the NATCONFG module.

Code Page Information

On the first screen, the following information is displayed:

Field	Explanation
*LOCALE	Shows the language and country of the current locale. See also system variable *LOCALE.
*CODEPAGE	Shows the IANA name of the code page currently used for conversions between Unicode and code page format, and is set by the profile parameter CP. See also system variable *CODEPAGE.
Source area CP	Shows the current code page of the source area. See also profile parameter CP.
CPOBJIN	Specifies the code page in which the batch input file for data is encoded. See also profile parameter CPOBJIN.
CPSYNIN	Specifies the code page in which the batch input file for commands is encoded. See also profile parameter CPSYNIN.
CPPRINT	Specifies the code page in which the batch output file shall be encoded. See also profile parameter CPPRINT.
CPCVERR	Specifies whether a conversion error that occurs when converting from U format to A format or from A format to U format or between A formats with different code pages results in a Natural error or not. See also profile parameter CPCVERR.
SRETAIN	Specifies that all existing sources have to be saved in their original encoding format. See also profile parameter SRETAIN.
CFICU	Enables Unicode support for various Unicode settings. See also profile parameter CFICU.

Field	Explanation
OPRB ACODE/WCODE	These values represent the key of supplied encoding for A fields and W fields during the user session for a given DBID. To change the DBID for OPRB ACODE/WCODE, press PF4. A window is displayed in which you can change the DBID for OPRB ACODE/WCODE.
CMPO CPAGE	Shows the current setting of the CPAGE option, which can be used to activate a conversion routine which translates all alphanumeric constants (from the code page that was active at compilation time into the code page that is active at runtime) when the object is started at runtime. See also keyword subparameter CPAGE of profile parameter CMPO.
WEBIO	Contains either the value of the system variable *BROWSER-IO or the value OFF. See also profile parameter WEBIO.
ICU version	Shows the current ICU version.
Unicode version	Shows the current Unicode version.
ICU module type	Shows the current implementation of the ICU library.

Press Enter to continue.

On the second screen, the following information is displayed:

Field	Explanation
Product	Product name: ICU for Adabas & Natural
Product code	ICS
Product version	ICS product version
Cumulated fix	Cumulated fix (if available and applied) for the current ICS product version
Architecture level	IBM architecture level The default is zero (0) denoting that no architecture level is used.
Revision	ICS revision number
ICU version	ICU version supported by ICS
Unicode version	Unicode version supported by ICS

Displaying Code Pages Defined in NATCONFG

Set the field Display code pages defined in NATCONFG (Y/N) to Y and press Enter.

Or press PF5 (NTCPG).

All code pages that are predefined in the macro NTCPAGE of source module NATCONFG are displayed.

Field	Explanation
Container Name	The internal ICU name (first 20 characters only).
IANA Name	The standard name of the code page (first 20 characters only).
CCSID	Coded Character Set Identifier. This identifier denotes the character set as identified by IBM.
First Alias Name	First defined alternate name for the code page (first 20 characters only).
D/E	This column shows the status of the code page in the source module `NATCONFG`. `E` Code page is defined in the `NATCONFG` module, and is enabled. `D` Code page is defined in `NATCONFG`, but is disabled.

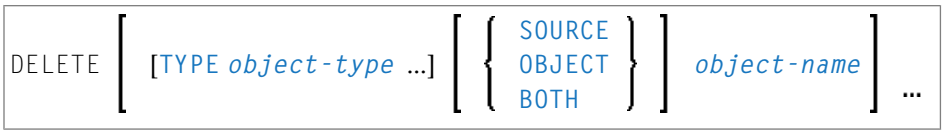
Further Information

See also:

- *Unicode and Code Page Support*
- *SYSCP Utility - Code Page Administration*

13 DELETE

- DELETE Syntax Explanation 64
- Selection List 65
- Safeguard Against Accidental Deletion 65
- DELETE Examples 65



This command is used to delete Natural objects from the Natural system file.

Note: The source currently in the editor's work area is not affected by the DELETE command.

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

DELETE Syntax Explanation

object-name	As *object-name*, you specify the name(s) of the object(s) to be deleted. In addition, you can specify whether only the source object, the corresponding cataloged object or both the source object and the cataloged object are deleted:	
	SOURCE	source object
	OBJECT	cataloged object
	BOTH	both source object and cataloged object. This is the default.
	A SOURCE/OBJECT/BOTH specification applies for all subsequent object names that is, until the next SOURCE/OBJECT/BOTH specification. To delete all objects whose names begin with a specific string of characters, you can use asterisk notation for the *object-name*.	
object-type	In conjunction with asterisk notation for the *object-name*, you can also specify an *object-type* if you wish to delete only objects of a specific type. The possible settings for *object-type* are the same as for the system command EDIT. In addition, you can specify the settings X (= global, local and parameter data areas) and U (= subprograms, subroutines and helproutines). **Note:** If you specify the full names of individual objects, you need not specify their object types.	

Selection List

If you use asterisk notation, you will get a selection list, on which you then mark the object(s) to be deleted. For each object, you can determine whether to delete the corresponding source object (**S**), the corresponding cataloged object (**O**) or both objects, by marking the object with the appropriate letter `S`, `O` or `B`, respectively.

If you enter only the `DELETE` command itself, you will also get a selection list, containing all objects stored in your current library.

Safeguard Against Accidental Deletion

As a safeguard against accidental deletion, a window will automatically be displayed in which you have to confirm the deletion of an object by entering its name.

If you have specified or selected more than one object, an additional window will be displayed in which you can specify whether you wish to confirm the deletion for each object individually or whether all specified/selected objects are to be deleted without confirmation.

DELETE Examples

With this command, you delete three objects named `TOM`, `DICK`, and `HARRY`:

```
DELETE TOM DICK HARRY
```

With this command, you delete the source object and the cataloged object `JOHN`, the source objects `PAUL` and `GEORGE`, and the cataloged object `RINGO`:

```
DELETE JOHN SOURCE PAUL GEORGE OBJECT RINGO
```

With this command, you get a selection list of all objects in the current library:

```
DELETE
```

With this command, you get a selection list of the source objects of all maps in the current library:

```
DELETE TYPE M SOURCE *
```

With this command, you get a selection list of all global, local and parameter data areas in the current library which are stored as source objects and/or catalgoged objects and whose names begin with D:

```
DELETE TYPE GLA D*
```

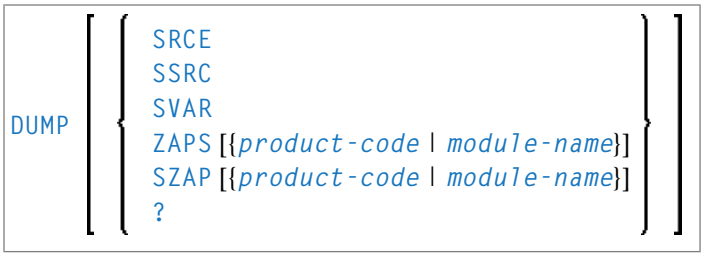
With this command, you get a selection list of all cataloged objects in the current library whose names begin with YYZ:

```
DELETE OBJECT YYZ*
```

With this command, you delete the source objects and cataloged objects of the maps TOM and DICK, the source object of the map HARRY, the source object of the program JOHN, and the cataloged object PAUL:

```
DELETE TYPE M TOM DICK SOURCE HARRY TYPE P JOHN OBJECT PAUL
```

14 DUMP



This command is used to provide information for support personnel in order to locate an error that caused an abnormal termination (abend) of the Natural system. Forward this information to support for error diagnosis and correction.

In the following table, the term "product" refers to base Natural and Natural add-on products. The list of source changes or Zaps that may be returned by a DUMP command refer to source changes and Zaps applied to the current product version after initial installation.

`DUMP`	Displays abend information (core contents).	
`DUMP SRCE`	Lists source changes applied to products linked to the current Natural nucleus.	
`DUMP SSRC`	Lists special source changes applied to products linked to the current Natural nucleus.	
`DUMP SVAR`	Displays TP monitor and operating system dependent system variables and additional information.	

`DUMP ZAPS`
 	
or
`DUMP SZAP`
	Lists Zaps (`DUMP ZAPS`) or special Zaps only (`DUMP SZAP`) applied to products linked to the current Natural nucleus.	

	`product-code`	Lists Zaps applied to a single product linked to the current Natural nucleus. *`product-code`* is the letter code that corresponds to the required product, for example, `NAT` for base Natural.

		All valid product codes are listed in *Overview of New Natural Add-On Product Versions* in the current Natural *Release Notes for z/OS*. There are three special product codes: `ASM` (for NATASMvr), `RSM` (for NATRSMvr), and `DRV`, which shows the ZAPs for the current TP driver that Natural is running on.
	`module-name`	Lists Zaps applied to a module that is not linked to the current Natural nucleus but concatenated to the Natural load library in the Natural execution JCL or in the appropriate library of the TP monitor program. *`module-name`* is a module name that has at least five characters. Support personnel will advise you of the required module name.

`DUMP ABEND`
 	
or
`DUMP ABND`
	Displays an overview of all important information after a program check (NAT0054) occurred, like Program name, Line number, registers, PSW, etc.	

`DUMP CSECTS`
 	
or
`DUMP CSCT`
	Shows all CSECTs in the nucleus.	

`DUMP EPLS`	Shows all 3-GL routines that have been linked to the nucleus, as well as the number of calls.	
`DUMP EPLN`	Like DUMP EPLS, but sorted by name.	
`DUMP EPLP`	Like DUMP EPLS, but sorted by program.	

`DUMP ?`
 	
or
`DUMP HELP`
	The `DUMP` command provides several other options (as explained on the help screens you get when you enter a question mark (?) on the `DUMP` menu). If necessary for error diagnosis, support personnel will tell you when and how to use these options.	

15 EDIT

- EDIT Syntax 1 70
- EDIT Syntax 2 72
- EDIT Syntax 3 72

This command is used to invoke a Natural editor for the purpose of creating and editing a Natural source.

Note: This command is not available if the Natural program, data area, and map editor are disabled in your environment. Use the source editor of NaturalONE to edit a source object.

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

Related command: READ.

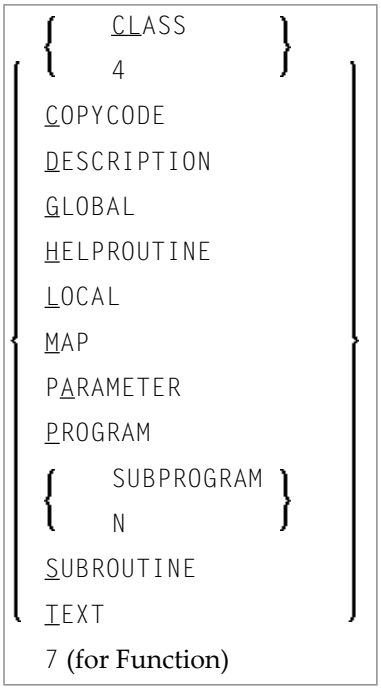
See also *Object Naming Conventions* in the *Using Natural* documentation.

EDIT Syntax 1

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

object-type

The following object types can be edited:



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.
- All other types of objects - program, subprogram, subroutine, 7 (for function), helproutine, copycode, text, description - are edited with the program editor.
- Descriptions are edited in Predict. 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 *Objects for Natural Application Management* in the *Programming Guide*. The editors are described in the *Editors* documentation.

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.

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.

Note: For EDIT DESCRIPTION, the *object-name* must be the name as defined as a Natural text object in the Predict program definition.

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.

Note: The setting for *library-id* must not begin with "SYS" (except SYSTEM).

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.

EDIT Syntax 2

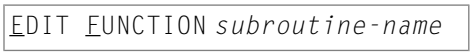


If you do not remember the name of the object you wish to edit, you can use this form of the `EDIT` command to display a list of objects, and then select from the list the desired object.

`EDIT *`	Displays a list of all objects in your current library.
`EDIT object-type *`	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 `LIST`.

EDIT Syntax 3



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
```

16 EDT

- EDT Syntax Explanation 74
- EDT Subcommands 74
- EDT Function Keys 75

This command is mainly intended to input Natural code lines into the source area, when running in a Batch environment without an input terminal. If the Natural session executes Online, we recommend that you use the EDIT command instead of the EDT command.

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

This command invokes the Natural line editor and causes edit mode to be entered. Edit mode may be used to edit an existing Natural object (program, copycode, subroutine, subprogram, helproutine). Once edit mode has been entered, you may position to any line and make changes, using the subcommands and PF keys listed below.

Use .E to terminate EDT mode.

EDT Syntax Explanation

object-name	As *object-name*, you enter the name of the object to be edited (maximum 8 characters). If *object-name* is entered, Natural will load the object into the source work area and set the object name for a subsequent SAVE, CATALOG, or STOW command. If you enter the EDT command without an object name and there is no object in the source work area, you will be prompted with line 0010 to enter an object. If you do not specify an object name and there is an object in the source work area, the first lines of that object will be displayed.
library-id	If the object to be edited is contained in a library other than the one to which you are currently logged on, you have to specify the ID of the library in which the object is contained. The setting for *library-id* must not begin with "SYS" (except SYSTEM). Entering a library ID is not permitted if Natural Security is active.

EDT Subcommands

The following subcommands may be used during line editing:

Command	Function
.B	Position to bottom.
.C*nnnn*(*m*)	Copy the line identified by *nnnn*. *m* indicates the number of lines to be copied.
.C'*text*'(*m*)	Copies the line that starts with *text*. *m* indicates the number of lines to be copied.
.D	Delete line.
.D(*n*)	Delete *n* lines.

Command	Function
.E	Exit from line editor.
.I	Insert line.
.I(*program*)	Insert *program*.
.M*nnnn*	Move the line identified by *n*.
.M'*text*'(*m*)	Move the line that starts with *text*. *m* indicates the number of lines to be moved.
.R	Renumber.
.S'*text*'	Scan for *text*.
.T	Position to top.
.*nnnn*	Position to line *nnnn*.
.+*n*	Position *n* lines forwards.
.-*n*	Position *n* lines backwards.

EDT Function Keys

To use PF keys during line editing, you must define the PF keys using the KEY profile parameter. You cannot assign EDT subcommands to a PF or PA key.

17 EXECUTE

- EXECUTE Syntax Explanation 78
- Examples of EXECUTE Command 79

```
{ 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.

EXECUTE Syntax Explanation

EXECUTE	The keyword EXECUTE is optional; it is sufficient to specify *program-name*, i.e. the name of the program to be executed. **Caution:** When entered in the command line of the program editor, the system command EXECUTE *must not* be abbreviated to EX, as the program editor would interpret this as the program editor command EX.
REPEAT	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.
program-name	The name of the program to be executed. If you do not specify a *library-id*, 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).
library-id	If the program is stored in another library, specify the *library-id* of that library. In this case, the program can only be executed if it is actually stored in the specified library. A *library-id* that begins with SYS must not be specified (except SYSTEM).
parameter	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). **Note:** The parameter values are always converted to upper case (regardless of the terminal command %L or the profile parameter LC=ON).

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
```

18 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 data set.

19 GLOBALS

- GLOBALS Syntax Explanation 84
- GLOBALS Parameters 84
- Interaction with SET GLOBALS and Other Statements 85

```
GLOBALS [parameter=value ...]
```

This command is used to set Natural session parameters.

GLOBALS Syntax Explanation

GLOBALS	If the GLOBALS command is entered without parameters, a screen appears where you can modify the parameter settings.
parameter	Parameter settings can be supplied in any order and must be separated by a blank. If more parameters are specified than will fit on one command line, several GLOBALS commands must be issued. Example: `GLOBALS DC=, ID=.`

GLOBALS 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
CPCVERR	Code Page Conversion Error
DC	Character for Decimal Point Notation
DFOUT	Date Format for Output
DFSTACK	Date Format for Stack
DFTITLE	Output Format of Date in Standard Report Title
DO	Display Order of Output Data
DU	Dump Generation
EJ	Page Eject
FCDP	Filler Character for Dynamically Protected Input Fields
FS	Default Format/Length Setting for User-Defined Variables
IA	Input Assign Character
ID	Input Delimiter Character
IM	Input Mode

Parameters	Function
LE	Reaction when Limit for Processing Loop Exceeded
LS	Line Size
LT	Limit for Processing Loops
MT	Maximum CPU Time
NC	Use of Natural System Commands
OPF	Overwriting of Protected Fields by Helproutines
PD	Limit of Pages for NATPAGE
PM	Print Mode
PS	Page Size for Natural Reports
REINP	Issue Internal REINPUT Statement for Invalid Data
SA	Sound Terminal Alarm
SF	Spacing Factor
SL	Source Line Length
SM	Programming in Structured Mode
THSEPCH	Thousands Separator Character
TS	Translate Output from Programs in System Libraries
WH	Wait for Record in Hold Status
ZD	Zero-Division Check
ZP	Zero Printing

Interaction with SET GLOBALS and Other Statements

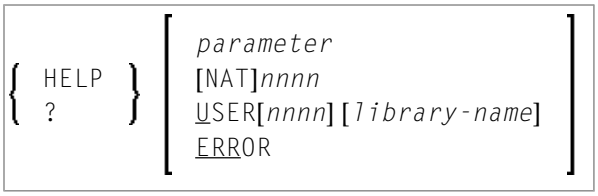
Statement SET GLOBALS

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.

20 HELP



This command is used to invoke the Natural Help utility. It retrieves information on Natural statements, commands, etc., and error messages.

For further information, see *Natural Online Help* in *Using Natural*.

HELP	Displays the help menu.	
HELP *parameter*	Displays help information for the specified *parameter*. Possible parameters are:	
	Parameter	**Command Examples**
	statement	HELP FIND HELP DEFINE WORK FILE
	system-variable	HELP *CODEPAGE
	system-function	HELP SORTKEY
	system-command	HELP LIST
	session-parameter	HELP AD
	terminal-command	HELP %<TEST
	Note: 1. Specify only the basic form of a statement, system variable, system command, etc. Valid example: HELP AD; invalid example: HELP AD=O. 2. Invalid commands will cause a parameter error display, for example, when you request help for a topic that is not supported: Parameter Error: SOSI.	

`HELP [NAT]`*`nnnn`*	Entering `HELP` 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 NAT*nnnn*.
`HELP NAT`	Displays information on all error messages.
`HELP USER`*`nnnn`*	Displays the long text of the library-specific error message number *nnnn* in the current library.
`HELP USER`*`nnnn`* `[`*`library-name`*`]`	Displays the long text of the library-specific error message number *nnnn* in the specified library.
`HELP USER`	Displays a selection list of *all* library-specific messages in the current library.
`HELP USER [`*`library-name`*`]`	Displays a selection list of *all* library-specific messages in the specified library.
`HELP ERROR`	Displays the long text for the error that occurred last.

21 INPL

```
INPL [R]
```

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

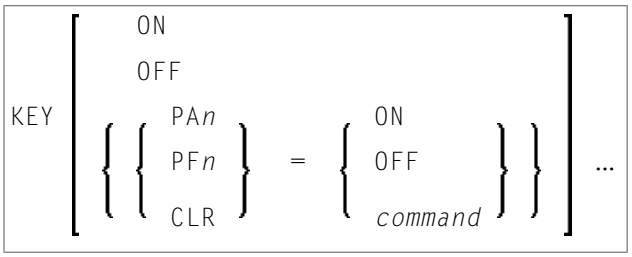
Apart from that, you use the Object Handler to load objects into the system files.

`INPL`	If you enter the `INPL` command without any parameters, the `INPL` utility will be invoked.
`INPL R`	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`: 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 *Inaccessible Security Profiles* in the section *Countersignatures* of the *Natural Security* documentation.

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

22 KEY

- Assigning Commands 92
- Activating/Deactivating All Keys - KEY ON/OFF 93
- Activating/Deactivating Individual Keys - KEY key=ON/OFF 93



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 (`NEXT prompt`).

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:

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

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.



Note: The CLEAR key cannot be activated/deactivated. Unless another function is assigned to it, it has the same function as the terminal command %%. The command KEY ON/OFF and the **Activate Keys** field have no effect on the CLEAR key.

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*.

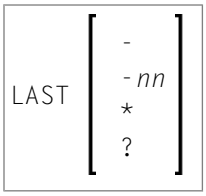
KEY *key*=OFF	Deactivates the command assigned to a specific *key*. For example: `KEY PF24=OFF`

`KEY` *key*`=ON`	Re-activates a previously deactivated command assignment. For example: `KEY PF24=ON`



Note: The commands `KEY CLR=ON` and `KEY CLR=OFF` are not possible (see also ***Note*** above).

23 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.

LAST	The command that was issued last is placed in the command line or NEXT line and can be executed.
LAST -	The command that was issued last is placed in the command line or NEXT line and can be executed. If you enter LAST - again, the last but one command is placed in the command line or NEXT line. By repeatedly entering LAST -, you can thus "page" backwards command by command. **Note:** Instead of repeatedly entering it by hand, you can assign LAST - to a PF key via the system command KEY.
LAST -*nn*	Natural "remembers" up to the last 20 commands that were issued; *nn* must therefore not be greater than 20. The last command but *nn* is placed in the command line or NEXT line and can be executed.
LAST *	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 *single* command again, either mark the command with the cursor and press F5, or mark the command with any character and press ENTER. ■ To execute *several* 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.
`LAST ?`	The Help function for the `LAST` command is called.

24 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, and the date and time behind the error message itself.

To display information on the corresponding error

- Mark one of these messages with the cursor and press ENTER or PF5.

  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.

Function Keys

The output screen of the LASTMSG command provides the following function keys:

PF Key	Name	Function
PF1	Help	Invoke help. Provides information about the use of PF2 and PF5.
PF2	Clear	Clears all the error messages from the error message collector.
PF5	Info	Provides detailed information about an error, when the cursor is placed on the error message and you press PF5. The same thing happens when the cursor is placed on an error message and you press ENTER.

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

Natural RPC (Remote Procedure Call):

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

25 LIST

- LIST Syntax Overview 100
- Listing the Contents of the Work Area 107
- Displaying an Individual Source Code 107
- Displaying a List of Objects 108
- Displaying a Presorted List of Preselected Objects 108
- Displaying Numbers and Sizes of Objects 108
- Displaying Sources Sequentially 109
- Displaying Directory Information 109
- Displaying Long Names of Cataloged Subroutines, Classes, and Functions 111
- Displaying NOC Options of Cataloged Objects 111
- Displaying Compiler Options of Cataloged Objects 111
- Displaying Compiler Output of a Cataloged Object 111
- Displaying DDMs (Views) 113
- Options 113
- List of Objects 117
- List of Source 125
- Defining an Individual List Profile 130

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. The options of the LIST command are explained below.

This chapter covers the following topics:

See also LIST XREF and the LIST commands specific to Db2 or SQL database management systems described in separate documentation sections.

Application programming interfaces: USR1054N, USR1055N, USR1056N, USR2018N, USR4216N. See *SYSEXT - Natural Application Programming Interfaces* in the *Utilities* documentation.

LIST Syntax Overview

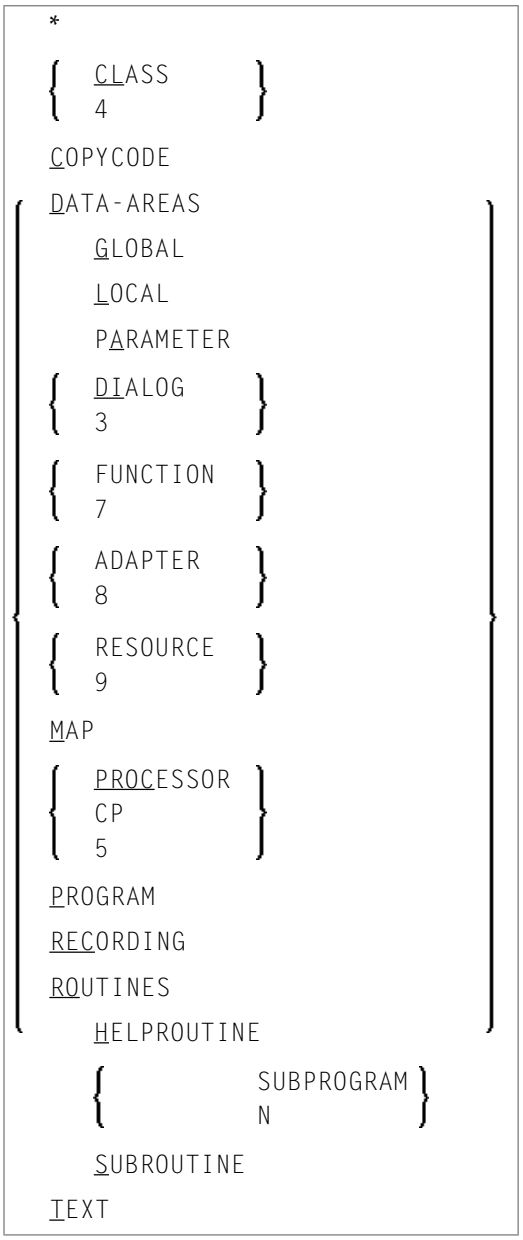
```
LIST [object-type] object-name-range [settings]
     [object-type] object-name [options] [settings]
     object-name-range [range-clause] [settings]
     COUNT [object-name-range] [range-clause] [settings]
     SEQUENTIAL [object-type] object-name-range [options]
     DIRECTORY [ object-name | object-name-range ]
     EXTENDED [ extended-type] object-name-range [settings]
     NOCOPT [object-type] object-name-range [settings]
     OPTIONS [object-type] object-name-range [settings]
     COMPOUT [compout-options] [settings]
     DDM [ddm-name]
```

Notes:

1. Instead of the keyword DDM, you can also use the keyword VIEW (or V for short).
2. Since LIST can display long lines containing up to 244 characters, set the line size as big as possible, using profile parameter LS. If possible, set LS=250.

object-type

In place of *object-type*, you may specify one of the object types shown below or an asterisk (*).



object-name

In place of *object-name*, you may specify the name of an object (8 characters long at maximum; exception: 32 characters with `LIST EXTENDED`).

object-name-range

In place of *object-name-range*, you may specify asterisk (*) and wildcard (?) notations:

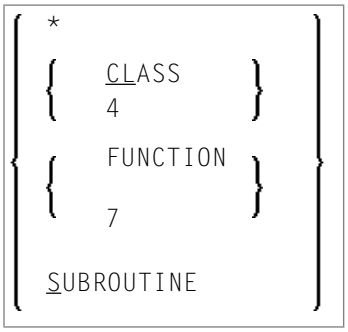
- To have all objects in the current library listed, you specify an asterisk (*) for the *object-name-range*, but no *object-type*.
- To have all objects of a certain type listed, you specify a certain *object-type* and an asterisk (*) for the *object-name-range*.
- If you wish a certain range of objects to be listed, you can use asterisk notation and wildcard notation for the *object-name-range*:
  - Asterisk notation is the option to specify an asterisk (*) in the *object-name-range*: the asterisk stands for any string of characters of any length.
  - Wildcard notation is the option to specify a question mark (?) in the *object-name-range*: the question mark stands for any single character.
- One or more asterisk and wildcard notations can be combined in an *object-name-range*.
- For a list of all objects from a specific start value or until a specific end value, you can use the notation > or < respectively.
- The notations < and > cannot be combined with each other or with asterisk or wildcard notation and can only be used for displaying a list of objects (see ***List of Objects*** below).

options

For a detailed description of the *options*, see ***Options***.

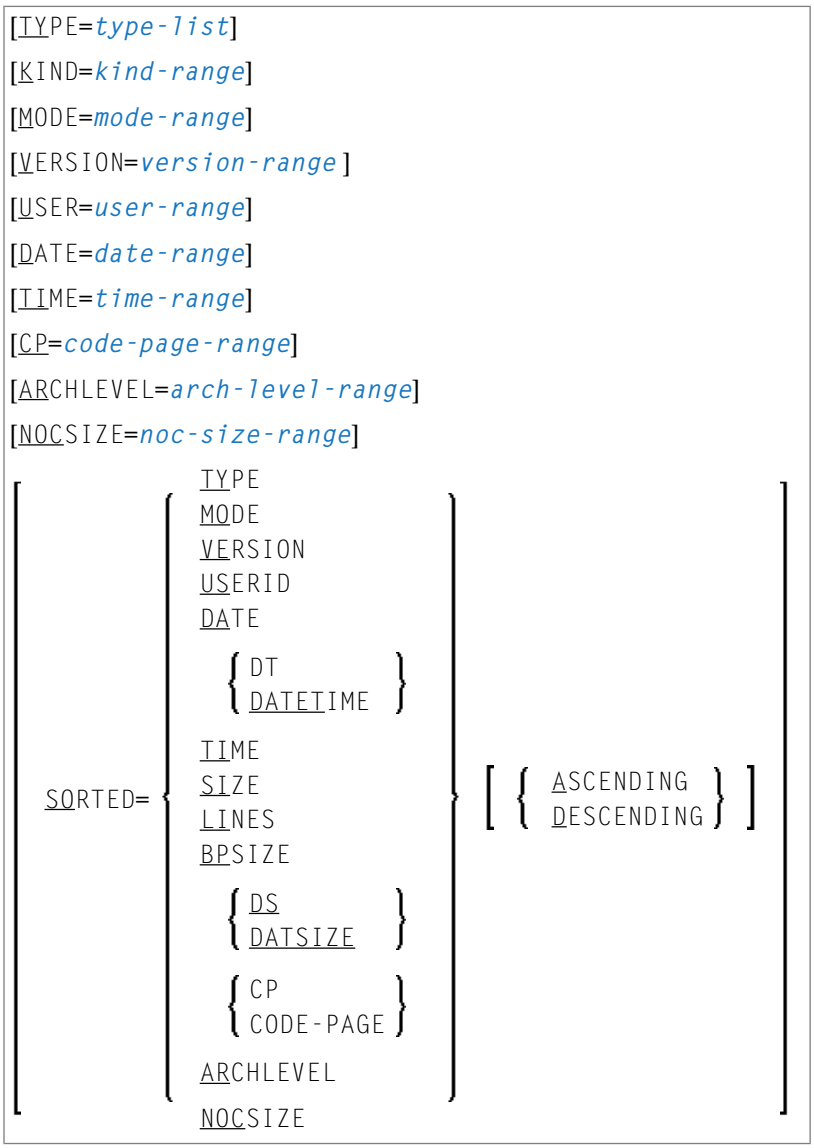
extended-type

In place of *extended-type*, you may specify one of the object types shown below or an asterisk (*).



For a detailed description, see LIST EXTENDED below.

range-clause



Notes:

1. SORTED cannot be used with the LIST COUNT command.
2. If you do not specify a direction for the SORTED parameter, the sort order will automatically be ascending.
3. See also the SORT subcommand in the section *Sorting the List of Objects*.

Syntax Element Description:

Syntax Element	Description	
type-list	* (for all types) or a list of up to 11 valid 1-byte Natural object type characters (e.g. P for Program, M for Map).	
kind-range	*	List all objects.
	S	List only source objects.
	C	List only cataloged objects.
	S/C	List only objects which exist as source and cataloged.
	S/	List only objects which exist as source only.
	/C	List only objects which exist as cataloged only.
	W	List only stowed objects.
mode-range	*	List all objects.
	S	List only structured mode objects.
	R	List only report mode objects.
version-range	The Natural version of the Natural objects. You can also specify a range of versions: see *range-notation*. See also the definition of the term Version in the *Glossary*.	
user-range	The ID of the user who saved or cataloged a Natural object. Specify a single user ID or a range of user IDs: see *range-notation*.	
date-range	Selects all objects with a save or catalog date within the date range specified. Specify a single date or a date range. Valid date format: *YYYY-MM-DD* Valid date ranges: ■ Leading characters (Example: 2002*) ■ Start value (Example: 2002-05>) ■ End value (Example: 2003-02<) Special dates allowed are:	
	TODAY (+/-*nnnn*)	All items with the date of the current day. The day can be followed by +*nnnn* or -*nnnn* where *nnnn* has a maximum of 4 digits. The resulting date is computed as the date of the current day plus or minus *nnnn* days. Can be combined with the start value option (>) or the end value option (<), e.g. TO-1> selects all objects that were saved or cataloged within the last 2 days.

Syntax Element	Description	
	YESTERDAY	All items with the date of the day before the current day.
	MONTH	All items with the date range of the current month.
	YEAR	All items with the date range of the current year.
time-range	Selects all objects with a save or catalog date within the time range specified. Specify a single time or a time range. Valid time format: *HH:II:SS* (*HH* = hours, *II* = minutes, *SS* = seconds). Valid time ranges: ■ Leading characters (Example: 10:*) ■ Start value (Example: 10:30>) ■ End value (Example: 11:20<)	
code-page-range	Specifies a single code page or a range of code pages: see *range-notation* below.	
arch-level-range	Only applies if Natural Optimizer Compiler is installed and if the ARCH option is set to supply IBM architecture levels (see the *Natural Optimizer Compiler* documentation). Lists all Natural objects that were cataloged under the Natural Optimizer Compiler with the ARCH option set. You can specify a single level or a range of levels as explained below, where *n* is a number:	
	n	Lists all objects with levels equal to *n*.
	n>	Lists all objects with levels equal to or greater than *n*.
	n<	Lists all objects with levels equal to or less than *n*.
	*n**	List all objects with levels from *n*0 to *n*9.
	nn	Lists all objects with levels equal to *nn*.
	nn>	Lists all objects with levels equal to or greater than *nn*.
	nn<	Lists all objects with levels equal to or less than *nn*.
noc-size-range	Only applies if Natural Optimizer Compiler is installed and NOC code was generated (see the *Natural Optimizer Compiler* documentation). Lists all Natural objects that are cataloged under the Natural Optimizer Compiler. The NOC size is shown in bytes. You can specify a single size or a range of sizes as explained below, where *nnn* is a number with maximum 7 digits:	
	nnn	Lists all objects with a NOC size equal to *nnn*.
	nnn>	Lists all objects with a NOC size equal to or greater than *nnn*.

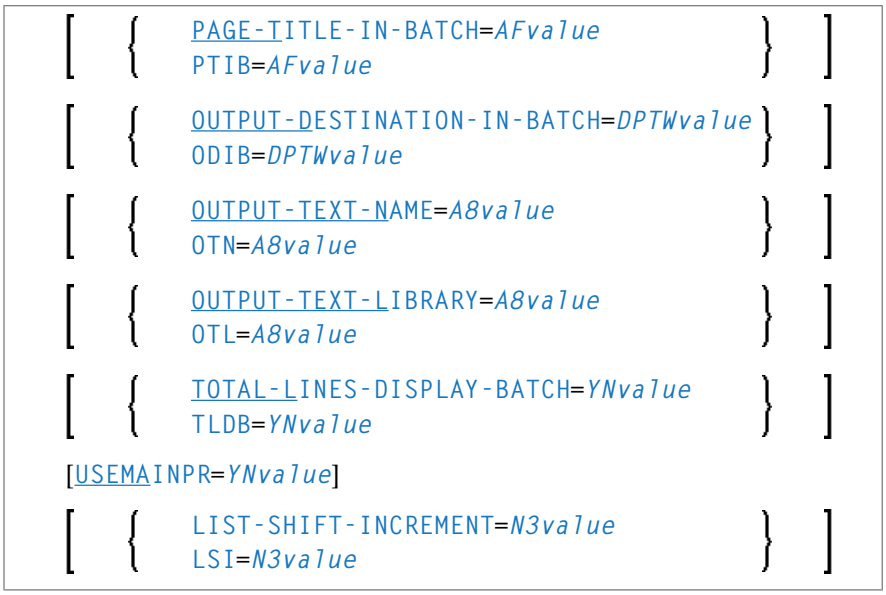
Syntax Element	Description	
	nnn<	Lists all objects with a NOC size equal to or less than *nnn*.

range-notation

- To have all objects in the current library listed, you use an asterisk (*).
- If you wish a certain range of objects to be listed, you can use asterisk notation and wildcard notation:
  - Asterisk notation is the option to specify an asterisk (*): the asterisk stands for any string of characters of any length.
  - Wildcard notation is the option to specify a question mark (?): the question mark stands for any single character.
- One or more asterisk and wildcard notations can be combined.
- For a list of all objects from a specific start value or until a specific end value, you can use the less than or greater than notation (< or >).
- The less than or greater than notation (< or >) cannot be combined neither with each other nor with asterisk (*) or wildcard (?) notation.

Settings

```
SET [ { REUSE-LAST-LIST=YNvalue }                    ]
    [ { RLL=YNvalue             }                    ]
    [ { COUNT-SOURCE-LINES=YNvalue }                 ]
    [ { CNTS=YNvalue               }                 ]
    [ { SORT-TEXT-MEMBER-NAME=A8value }              ]
    [ { STMNA=A8value                 }              ]
    [ { SORT-TEXT-MEMBER-LIBRARY=A8value }           ]
    [ { STMLI=A8value                     }          ]
    [ { DELETE-SORT-TEXT-MEMBER=YNvalue }            ]
    [ { DELST=YNvalue                   }            ]
    [ { PRINT-PAGE-SIZE=N3value }                    ]
    [ { PPS=N3value             }                    ]
    [MARK-LONG-LINES=YNvalue]
    [ { DATA-AREA-DEFINE-DATA-FORMAT=YNvalue }       ]
    [ { DADDF=YNvalue                        }       ]
    [ { SOURCE-LIST-WITH-DBID-FNR=YNvalue }          ]
    [ { SLDF=YNvalue                      }          ]
```



Where:

YNvalue= Y or N
N3value = 3-byte numeric value
A8value = 8-byte alphanumeric value
AFvalue = A or F
DPTWvalue = D, P, T or W

Note: For the description of the options, see ***LIST Profile Parameters***.

Listing the Contents of the Work Area

LIST	If you enter only the LIST command itself, without any parameters, the contents of the work area will be listed.

Displaying an Individual Source Code

LIST *object-name* [*options*]	In both cases, the object's source code will be listed. If you enter a single object name with the LIST command, you need not specify the *object-type*. If you specify an *object-type*, you must also specify an *object-name*.
LIST *object-type object-name* [*options*]	

Displaying a List of Objects

`LIST object-name-range` `LIST object-type object-name-range`	In both cases, you must use asterisk (*) and/or wildcard (?) notation for the *object-name-range*. You get a list of all objects that meet the specified selection criteria. On the list you can then select objects for display by marking them with the function code `LI` (see ***Performing a Function on an Object***).

Displaying a Presorted List of Preselected Objects

`LIST object-name-range` `LIST object-name-range range-clause`	In both cases, you must use asterisk (*) and/or wildcard (?) notation for the *object-name-range*. You get a list of all objects that meet the specified selection criteria. On the list you can then select objects for display by marking them with the function code `LI` (see ***Performing a Function on an Object***). With the *range-clause*, you specify additional selection and sorting criteria. See also **example** below.

Displaying Numbers and Sizes of Objects

`LIST COUNT [range-clause] [settings]`	Displays a summary report that contains the numbers and sizes (in bytes or KB if greater than 1 MB) of objects stored in the current library. The numbers and sizes listed refer to all objects that have been saved as source (saved) objects only or as cataloged objects only, and all objects for which both saved and cataloged objects (stowed) exist, according to the selection criteria specified in the *range-clause*. See also ***Examples of List of Objects Usage***.
`LIST COUNT object-name-range [range-clause] [settings]`	Displays an extended report where the numbers and sizes of saved/cataloged objects (same as above) are listed per object type(s) found, according to the selection criteria specified in the *object-name-range* and the *range-clause*. Object type `unknown` indicates that no directory information exists for these objects. See also ***Examples of List of Objects Usage***.

Displaying Sources Sequentially

`LIST SEQUENTIAL` *`object-name-range`* `[`*`options`*`]` `LIST SEQUENTIAL` *`object-type object-name-range`* `[`*`options`*`]`	In both cases, you must use asterisk (*) and/or wildcard (?) notations for the *`object-name-range`*. Then the sources of all objects that meet the selection criteria will be displayed sequentially, i.e. one after the other.

Displaying Directory Information

`LIST DIRECTORY`	Displays the directory information on the object currently in the work area: **Source code:** ■ Date/time when the source was saved, ■ library name, ■ user ID, ■ programming mode (reporting or structured), ■ TP system, ■ terminal ID, ■ operating system, ■ transaction code, ■ Natural version, ■ code page information (if available), ■ source size **Object code:** ■ Date/time when the object was cataloged, ■ library name, ■ DBID/FNR of library, ■ user ID, ■ programming mode, ■ TP system, ■ terminal ID, ■ transaction code, ■ Natural version,

	■ code page information (if available), ■ operating system, ■ size in DATSIZE, ■ size in buffer pool, ■ size of global data, ■ size of NOC code (machine code generated by Natural Optimizer Compiler), ■ GDA name used by the object, ■ date/time when the used GDA was cataloged, ■ depending on the usage of compilation option GDASC during cataloging: ■ either the comparison of the GDA signatures of the object and the used GDA (like identical, not identical) (see the **GDASC** documentation), ■ or, GDA time stamp from the date/time when the GDA was cataloged and when the object was cataloged, ■ number of subroutines used by the object, ■ architecture (ARCH) level used when cataloging the object with the Natural Optimizer Compiler (NOC) and the ARCH option set, ■ compiler options (initial, final, changed) when the object was cataloged, ■ Natural Optimizer Compiler (NOC) options set with the OPT profile parameter when the object was cataloged ■ long name of cataloged subroutine, function or class, ■ list of external subroutines used by the object (long name and name of cataloged object).
LIST DIRECTORY *object-name*	Displays the directory information (as described for LIST DIRECTORY) on the specified object. If asterisk (*) and/or wildcard (?) notation is used in place of *object-name-range*, the directory information of the corresponding objects is displayed sequentially.
LIST DIRECTORY *object-name-range*	
LIST *object-name* WITH DIRECTORY	This command first displays basic directory information (such as LIST DIRECTORY) on the specified object and then lists the source code of the object.

Displaying Long Names of Cataloged Subroutines, Classes, and Functions

`LIST EXTENDED` *object-name-range*	Displays a list of the long names of cataloged subroutines, classes and functions. For the name options, see *object-name-range* above.
`LIST EXTENDED` *extended-type object-name-range*	

Displaying NOC Options of Cataloged Objects

`LIST NOCOPT [`*object-type*`]` *object-name-range*	Displays a list of the cataloged objects that are compiled with Natural Optimizer Compiler (NOC), together with the initial NOC options used during `CATALOG`. For the name options, see *object-name-range* above.

Displaying Compiler Options of Cataloged Objects

`LIST OPTIONS [`*object-type*`]` *object-name-range*	Displays a list of the cataloged objects together with the compiler options used during `CATALOG`. For the name options, see *object-name-range* above. By default, the final compiler options (that is, the options setting active at the end of the `CATALOG`) are displayed. For objects cataloged with Natural Version 4.2.5 or above, also the initial compiler options (that is, the options setting active at the beginning of the `CATALOG`) or the changed compiler options (that is, the options setting that are changed within the source code) can be displayed. See the corresponding help map for the range fields on the online map.

Displaying Compiler Output of a Cataloged Object

`LIST COMPOUT` displays compiler information from the cataloged object in a report that can be displayed on screen or written to work file 1 or to a Natural source of type `TEXT`. If no additional options are specified or if required options are missing in the command, a map is displayed where the desired options and the output destination can be specified.

Command Syntax:



Syntax Element Description:

Syntax Element	Description
object-name	The name of the cataloged Natural object.
REPORT	Just a filler to make the command more readable, can be omitted.
UNVARIABLES	Add to the report all unused variables, that is, variables that are defined but not referenced or modified.
USEDVARIABLES	Add to the report all used variables, that is, variables that are referenced or modified.
CONTSUBROUTINES	Add to the report the contained external subroutine (for Natural objects of type SUBROUTINE only).
EXTSUBROUTINES	Add to the report the used external subroutines.
STATPROFILE	Creates a report that displays contiguous sequences of statements grouped by categories in a source program suitable for optimization, or lists the NOC coding generated for an optimized program; see also *NOCSTAT Command* and *Code Profile* in the *Natural Optimizer Compiler* documentation).
STATCATEGORY	Creates a report that lists various categories of statements with the corresponding number of occurrences; see also *NOCSTAT Command* and *Statement Category* in the *Natural Optimizer Compiler* documentation).
STATTYPE	Creates a report that lists single statements with the corresponding number of occurrences; see also *NOCSTAT Command* and *Statement Type* in the *Natural Optimizer Compiler* documentation).
DESTINATION	Just a filler to make the command more readable, can be omitted.
SCREEN	Show the report on the screen. This is the default value.
WORKFILE	Write the report to work file 1.
TEXT	Write the report to a Natural source of type TEXT.
text-name	Name of the Natural source of type TEXT. The default value is REPCPOUT.

Syntax Element	Description
LIBRARY *library-name*	Library name for the Natural source of type TEXT. The default value is the current library.

See also *Examples of List of Objects Usage*.

Displaying DDMs (Views)

LIST DDM	Displays a list of all DDMs.
LIST DDM *ddm-name*	If you specify a single DDM name, the specified DDM will be displayed. For the *ddm-name* you can use a single DDM name (up to 32 characters) or a range as for *object-name-range* to display a list of a certain range of DDMs.

Note: Instead of the keyword DDM, you can also use the keyword VIEW (or V for short).

Options

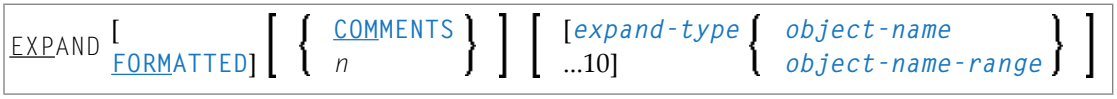
In place of *options*, you may specify one of the options shown below.

```
{ [[WITH] DIRECTORY] [NUMBERS OFF] [expand-option]
  formatted-option
  CONVERTED
  UNCONVERTED                                      }
```

Syntax Element Description

DIRECTORY	This option first displays basic directory information (such as *Displaying Directory Information*) on the specified object and then lists the source code of the object.
NUMBERS OFF	By default, the source code of an object will be listed with source-code line numbers. To list it without line numbers, specify the NUMBERS OFF option. (See also subcommands NUMBERS ON/NUMBERS OFF in the section *Subcommands for Listed Source*.)
UNCONVERTED	By default, the source is listed in the session code page (see system variable *CODEPAGE)). To list the source in the code page as stored on the system file, specify this option.
CONVERTED	If the source is listed in the code page as stored on the system file (by option or command UNCONVERTED), specify this option to list the source in the session code page (see system variable *CODEPAGE)).

expand-option



Syntax Element Description

`EXPAND` *object-name*
`EXPAND` *object-name-range*
	With the `EXPAND` option, you can have the sources of other objects referenced by the listed object - copycodes, data areas, maps, helproutines, external subroutines, subprograms, programs called with a `FETCH` statement, error messages - listed *within* the source of the listed object. This option is particularly useful in batch mode.
	
	For example, if a listed source program contains an `INCLUDE` statement, you can have the source code of the included copycode listed within the listed source program immediately after the `INCLUDE` statement.
	
	Objects listed within a source will be referred to as "expand objects" in the explanations below.
	
	Subcommands in Expand Object
	
	Within a listed expand object, only the following subcommands are available:
	
	`PRINT`
	`+`
	`- -`
	`.`
	
	See ***Examples of List of Objects Usage***.

`EXPAND FORMATTED`	The `EXPAND FORMATTED` option is only relevant for stowed data areas (where time stamp of source object and cataloged object are identical) and maps listed within a source. For data areas, the following applies: ■ If `FORMATTED` is not specified, the display of the data area will resemble that in the data area editor ■ If `FORMATTED` is specified, the display of the data area will resemble a `DEFINE DATA` statement. This only applies to stowed data areas (i.e. the time stamp of source object and cataloged object are identical); see also subcommand `FORMAT`. For maps, the following applies: ■ If `FORMATTED` is not specified, the map *source* will be listed. ■ If `FORMATTED` is specified, the map *layout* will be displayed (that is, the map as it is displayed to the users at runtime).

<table>
<tr><td>EXPAND COMMENTS
EXPAND n</td><td colspan="3">If you use the option EXPAND COMMENTS, only the initial comment lines of the expand object will be listed; that is, the expand object will be listed until (but not including) the first source-code line which is not a comment line.

If you use the option EXPAND n, only the first n lines of the expand object will be listed.

If you use neither of these two options, the entire expand object will be listed.</td></tr>
<tr><td>expand-type</td><td colspan="3">As expand-type, you specify the object type(s) of the expand object(s). The following expand-types can be specified:</td></tr>
<tr><td rowspan="11"></td><td>P</td><td>Programs</td><td rowspan="11">If you wish to specify more than one expand-type, you can specify them in any sequence and without blanks between them; for example, to have maps, copycodes and subroutines listed within the listed source, specify the expand-type as MCS.</td></tr>
<tr><td>N</td><td>Subprograms</td></tr>
<tr><td>S</td><td>External subroutines</td></tr>
<tr><td>H</td><td>Helproutines</td></tr>
<tr><td>G</td><td>Global data areas</td></tr>
<tr><td>L</td><td>Local data areas</td></tr>
<tr><td>A</td><td>Parameter data areas</td></tr>
<tr><td>M</td><td>Maps</td></tr>
<tr><td>C</td><td>Copycodes</td></tr>
<tr><td>E</td><td>Error messages</td></tr>
<tr><td>4</td><td>Classes</td></tr>
<tr><td>*</td><td>All object types</td></tr>
<tr><td>object-name
object-name-range</td><td colspan="3">As object-name or object-name-range, you specify the name(s) of the expand object(s) to be listed within the main listed source.

The same notations are possible as for object-name or object-name-range, except < and >.</td></tr>
</table>

formatted-option



FORMATTED Option

The FORMATTED option applies to stowed data areas (where time stamp of source object and cataloged object are identical) and maps:

FORMATTED	**Stowed Data Area:** If you specify the FORMATTED option for a data area, the data area will be displayed formatted; that is, the display resembles a DEFINE DATA statement; see also subcommand FORMAT. This only applies to stowed data areas (i.e. the time stamp of source object and cataloged object are identical). By default, data areas are displayed unformatted; i.e. the display resembles that in the data area editor. The default setting can be changed with in the List Profile. (Refer to ***Defining an Individual List Profile*** below and see also subcommand FORMAT). **Map:** If you specify the FORMATTED option for a map, the map *layout* will be displayed; that is, the map as it is displayed to the users at runtime.

FORMATTED Options for Listing Maps

When you are listing maps, you may specify options in addition to the keyword FORMATTED:

['*c*'] ['*c*']	**Using Filler Characters:** You may specify filler characters *c* for input fields (AD=A and AD=M) and output fields (AD=O) to make these fields visible. You may specify any character as filler character. The following example shows all input fields with an underscore (_) and all output fields with a hash (#). `LIST MAP map-name FORMATTED '_' '#'`
SETTINGS	**Map Settings:** Displays the map settings of the specified map. `LIST MAP map-name FORMATTED SETTINGS`
FIELDS	**Field Summary:** Displays the field summary; that is, the list of fields in the specified map. `LIST MAP map-name FORMATTED FIELDS`

EXTFIELDS	**Extended Field Editing Information:** Causes the extended field editing information for all map fields to be displayed. `LIST MAP map-name FORMATTED EXTFIELDS`

Diplaying Processing Rules for a Map

The following options cause the processing rules used by the map to be displayed. The rules are displayed in order of fields to which they are assigned, and per field in order of rank.

RULES	**All Rules:** `LIST MAP map-name FORMATTED RULES` Displays *all* the rules for the specified map.
INLINERULES	**Inline Rules Only:** `LIST MAP map-name FORMATTED INLINERULES` Displays only the inline rules for the specified map.
FREERULES	**Free Rules Only:** `LIST MAP map-name FORMATTED FREERULES` Displays only the free rules for the specified map.
AUTORULES	**Automatic Rules Only:** `LIST MAP map-name FORMATTED AUTORULES` Displays only the automatic rules for the specified map.

See also the subcommands LAYOUT and FORMAT in the section ***List of Source***.

List of Objects

When you use asterisk or wildcard notation for the object name, you get a list of all objects that meet the specified selection criteria. On this list, you can then select objects for display, print, etc. by marking them with a function code, or you can enter a Natural system command or a LIST subcommand in the command line.

This section describes the functions, subcommands and function codes that are available in the list of objects which is displayed, for example, after you have issued a `LIST *` command.

The following topics are covered:

- Explanation of the Column Headers
- Scrolling the Selection List of Objects
- New Criteria for the Selection List
- Information Displayed on the Selection List
- Items Intensified on the Selection List
- Subcommands for the Selection List
- Performing a Function on an Object
- Sorting the List of Objects
- Examples of List of Objects Usage

Explanation of the Column Headers

The list of objects contains the following columns:

Column	Explanation
`Cmd`	In this column, you can enter a code to perform a function on an object in the selection list. See *Performing a Function on an Object*.
`Name`	Name of object.
`Type`	Type of object.
`S/C`	Indicates whether the object exists as source (S) and/or cataloged object (C).
`SM`	The Natural programming mode that was used when the object was created. S = structured mode, R = reporting mode.
`Version`	Product version of Natural that was used to create or catalog the object.
`User ID`	User ID of the user who created or cataloged the object.
`Date, Time`	Date and time when the object was created or cataloged.

Scrolling the Selection List of Objects

Once a list of objects is displayed, you can scroll it as follows:

- To scroll the list one page forward or backward, press PF8 or PF7 respectively.
- To scroll the list to its beginning or end, press PF6 or PF9 respectively.

New Criteria for the Selection List

When a list of objects is displayed, the fields immediately underneath the column headings show the selection criteria for the current list. You can change the selection criteria by overwriting the values of these fields. For information on the possible values for one of these fields, you enter a question mark (?) in the field.

Information Displayed on the Selection List

If there exists both a source and an object module for an object (as indicated in the column **S/C**), the information displayed refers to the source, not the object module.

Note: When the sort function is active the source and the object module may be displayed separately, e.g. when the list is sorted by the object date and the source and the object module have different date values.

To display more information on source and cataloged objects

- Press PF11 to shift right.

  Or:

  Press PF10 to shift left.

Note: By default the number of source lines of source objects is not calculated due to performance reasons. If you want the number of source lines of source objects being displayed, you can either enter the subcommand `COUNTSOURCE ON` or set in the `LIST` profile (see *Defining an Individual List Profile* below) the parameter `COUNT-SOURCE-LINES` to `Y`.

Items Intensified on the Selection List

If an item is displayed intensified on the left-most list page, this indicates that there is a discrepancy between the object's source and its object module. For information on the discrepancy, you may mark the object with the function code `LD` to list its directory information. To eliminate the discrepancy, it is usually sufficient to stow the object again (function code `ST`).

Subcommands for the Selection List

In a list of objects, you can enter a Natural system command or a `LIST` subcommand in the command line. Valid subcommands are:

Code	Function
`SC`	List only objects containing a scan value (can only be used with long list).
`SC OFF`	Switch off scan mode.
`SHORT`	Display a short list of objects, i.e., display only the object names (can only be used if scan mode is off).
`LONG`	Switch to long list including all fields available.
`PRINT`	Print the list of objects.
`EXTENDED`	Display the list of long names of subroutines, classes, and functions; same as `LIST EXTENDED *`.

Code	Function	
ALL *fx*	Enter the function code *fx* (where *fx* is a valid function code for a listed object) for all displayed objects.	
SORT	Invoke the sort window (see ***Sorting the List of Objects*** below).	
COUNTSOURCE	ON	Display the number of source lines for source objects.
	OFF	Do not display the number of source lines for source objects.
MARK-LONG-LINES	ON	Mark long lines in the list of a source object with an L in the first two positions. The default value can be specified in the LIST profile; see ***Defining an Individual List Profile***.
	OFF	Do not mark long lines in the list of a source object.
DEFINE-DATA	ON	A listed data area source is listed in DEFINE DATA format by default (same as LIST *data-area* FORMATTED). The default value can be specified in the LIST profile; see ***Defining an Individual List Profile***.
	OFF	A listed data area source is listed unformatted.
LISTCOUNT	List the number of objects and their sizes according to the selection criteria currently used.	
LISTPROFILE	Display the current value of the parameters of the LIST profile (see ***Defining an Individual List Profile*** below).	
NOCOPT	Display a list of the cataloged objects that are compiled with Natural Optimizer Compiler (NOC), together with the initial NOC options used during CATALOG; same as LIST NOCOPT *, see ***Displaying NOC Options of Cataloged Objects***.	
OPTIONS	Display a list of the cataloged objects together with the initial compiler options used during catalog; same as LIST OPTIONS *, see ***Displaying Compiler Options of Cataloged Objects***.	
USE-TIMESTAMP	ON	Use the time stamp of the cataloged object for displaying the catalog date and time in the object list.
	OFF	Use the date and time values from the cataloged object directory for displaying the catalog date and time in the object list. See also the option SHOW-CAT-TIMESTAMP-VALUES in *Defining an Individual List Profile*.
REUSE	ON	Switch on reuse mode. The last displayed list is reused after execution of commands entered in the **Cmd** column, except for the following commands: E ED (Edit)

Code	Function	
		CA (Catalog) UC (Uncat) S ST (Stow) D DE (Delete) RE (Rename)
	OFF	Switch off reuse mode. The list is rebuilt after execution of commands entered in the **Cmd** column.
REFRESH	Rebuild the currently displayed list. This subcommand can be used especially when reuse mode is switched on.	
+	Scroll one page forward.	
-	Scroll one page backward.	
++	Scroll to the end (bottom) of the object list.	
--	Scroll to the beginning (top) of the object list.	
?	Command line help.	

Performing a Function on an Object

To perform a function on an object in the selection list, you simply mark the object with the appropriate function code in the left-hand column (titled **Cmd**).

You can mark several objects on the selection list with different function codes; the functions will then be performed one after the other.

The following function codes are available (possible abbreviations are underlined).

Code	Function
?	A window will be displayed which shows all the functions available for the marked object. The window will only list those functions that are actually available for the selected object (for example, if the object is a subroutine, it cannot be run; if the object is only available in source form, it cannot be executed). From the window you can select the function to be performed on the marked object.
CA	Compile the object and store it in object form (equivalent to the system command CATALOG).
DE	Delete the object (equivalent to the system command DELETE).
DL	Download object from z/OS to personal computer (only available if Natural Connection is installed).

Code	Function
ED	Edit the object's source code (equivalent to the system command EDIT). **Note:** This command is not available if the Natural program, data area, and map editor are disabled in your environment. For more information, see *NaturalONE as the Default Development Environment* in the *Editors* documentation.
EX	Execute the object (equivalent to the system command EXECUTE).
LC	List object's source code converted into the default code page *CODEPAGE, (equivalent to LIST *object-name* CONVERTED).
LD	List directory information (equivalent to LIST DIRECTORY *object-name*) on the object.
LE	List object's source code in expanded form (equivalent to LIST *object-name* EXPAND *).
LF	Display a data area or map formatted (equivalent to LIST *object-name* FORMATTED).
LI	List the object's source code.
LN	Display long name of subroutine or class (only possible if a cataloged object exists) or resource.
LO	Invoke LIST COMPOUT for the cataloged object (equivalent to LIST COMPOUT *object-name*); see ***Displaying Compiler Output of a Cataloged Object***.
LU	List object's source code converted into code page as stored on the system file (equivalent to LIST *object-name* UNCONVERTED).
NO	Displays the Natural Optimizer Compiler (NOC) options used during CATALOG (only possible if a catalog object exists).
OP	Displays the initial, final and changed Natural compiler options used during CATALOG (only possible if a cataloged object exists). The initial and changed compiler options can be displayed for objects cataloged with Natural Version 4.2.5 or above only.
PR	Print the object's source code.
RE	Rename the object (equivalent to the system command RENAME).
RU	Run (that is, compile and execute) the object's source code (equivalent to the system command RUN).
ST	Stow the object in source and object form (equivalent to the system command STOW).
UC	Delete the object module (equivalent to the system command UNCATALOG).
.	Exit (from selection list window)

Sorting the List of Objects

The LIST command provides the possibility to sort the list of the displayed objects by several sort criteria.

Note: To use this function, it is necessary to set the WRKSIZE (Size of Work Buffer Used by Sort Program) in the Natural profile parameter SORT to an appropriate value. The maximum size of the list that can be sorted is limited by the size of this work buffer.

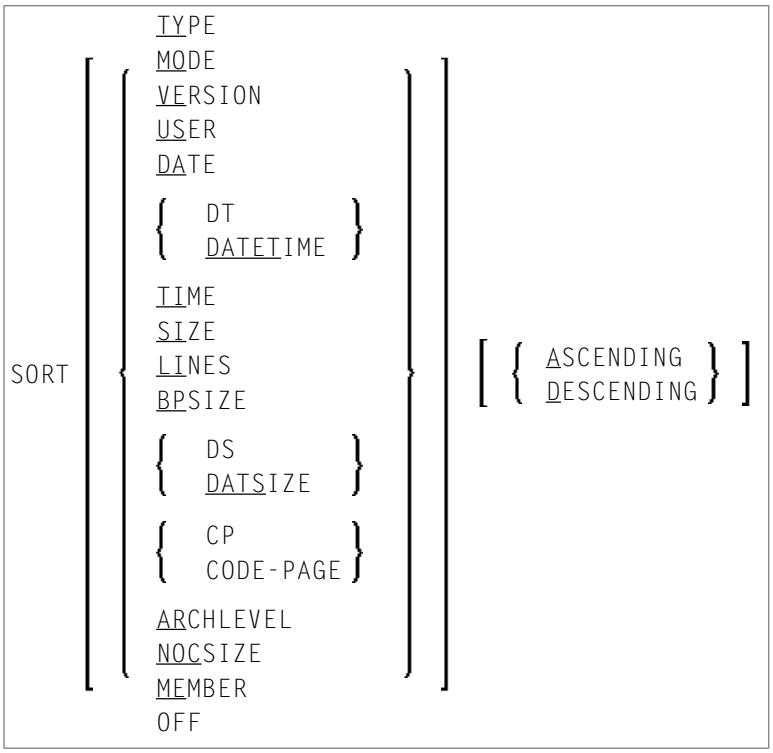
To invoke the sort function

- Press PF4.

  Or:

  Enter a SORT subcommand on the list of objects.

SORT Subcommand Syntax



When you press PF4, a window is displayed where you can specify whether you want to sort the list or the sort field, and the sort order. You can sort the list in ascending or descending order by the following sort fields. The default order is ascending.

Sort Field	Keyword in Sort Syntax
Natural object type	TYPE
Programming mode (reporting or structured mode)	MODE
Version	VERSION
User ID	USER
Date	DATE
Date and time	DATETIME
Time	TIME

Sort Field	Keyword in Sort Syntax
Source size	SIZE
Number of source lines	LINES
Buffer pool size	BPSIZE
DATSIZE (size of local data buffer)	DS/DATSIZE
Code page	CP/CODE-PAGE
ARCH level	ARCHLEVEL
NOC size	NOCSIZE
Member names of subroutines, classes, or functions (available in extended selection list only)	MEMBER

If you change the selection criteria after a list has been sorted and displayed, this will result in the creation of a new list from all objects contained in the library according to the selection criteria, which is then sorted by the selection criteria.

To switch off the sort mode

- Enter the subcommand SORT OFF.

  Or:

  Deactivate the sort function in the **Sort Options** window invoked by pressing PF4.

The sorted list is built in a Natural text object in library WORKPLAN. The name of the text object is generated by the LIST command. If the LIST profile is activated (see *Defining an Individual List Profile* below) the name of the text object and the library can be specified in the LIST profile.

Examples of List of Objects Usage

LIST *	Lists all objects in the current library.
LIST S *	Lists all subroutines in the current library.
LIST SYS*	Lists all objects (of any type) whose names begin with SYS.
LIST M SYS*	Lists all maps whose names begin with SYS.
LIST C *CODE	Lists all copycodes whose names end with CODE.
LIST NAT*AL	Lists all objects whose names begin with NAT and end with AL no matter which and how many other characters are between NAT and AL (this would include the names NATURAL and NATIONAL as well as NATAL).
LIST DOO?	Lists all objects with 4-character names beginning with DOO (this would include the names DOOR and DOOM, but not DOO or DOODLE).
LIST M NAT?AL	Lists all maps whose names begin with NAT and end with AL with exactly one character are between NAT and AL (this would include the names NAT1AL and NAT2AL, but not NATAL or NATIONAL).

`LIST M *1*`	Lists all maps whose names contain a `1`.
`LIST M F>`	Lists all maps, starting from the first one whose name begins with `F`.
`LIST M MA<`	Lists all maps, from the first one until the one named `MA` (if present).
`LIST N?T*AL`	Lists all objects such as `NATAL`, `NATURAL`, `NAT`*`vr`*`AL` (where *`vr`* represents the relevant product version).
`LIST E* TYPE=PM KIND=S DATE=YEAR SORTED=DATE ASCENDING`	Creates a list of all source objects of programs and maps whose names start with `E` and which were saved in the current year. The list is sorted by object date in ascending order.
`LIST COUNT`	Lists the total number/sizes of all objects in the current library for which saved and/or cataloged objects exist.
`LIST COUNT * TYPE=PM KIND=S`	Lists the numbers/sizes of all programs and maps for which saved objects exist.
`LIST COMPOUT HUGO REPORT UNV EXTS WORKF`	Writes a report of the cataloged object `HUGO` to Work File 1 with all unused variables and all used external subroutines.

List of Source

The following topics are covered below:

- Subcommands for Listed Source
- Subcommands FIND, REF and SCAN
- Subcommand FORMAT
- Cursor-Sensitive Object Selection

Subcommands for Listed Source

When you have the source code of an object listed, in the command line, you can enter one of the subcommands listed in the following table.

Note: By default, the database id (DBID) and file number (FNR) of the source library are not displayed in the header line of the listed source. If you want the DBID and FNR of the source library to be displayed, you can either enter the subcommand `DBFNR ON` or, in the `LIST` profile, set the parameter `SOURCE-LIST-WITH-DBID-FNR` to `Y` (see ***Defining an Individual List Profile*** below).

Subcommand	Function
`+`	Scrolls one page forward.
`-`	Scrolls one page backward.
`+H`	Scrolls forward half a page.
`-H`	Scrolls backward half a page.
`++`	Scrolls to the end (bottom) of the source.

Subcommand	Function
BOTTOM	
--	Scrolls to the beginning (top) of the source.
TOP	
+*n*	Scrolls *n* lines forward.
-*n*	Scrolls *n* lines backward.
nnnn	Scrolls to line number *nnnn*.
UNCONVERTED	See UNCONVERTED in *Options*.
CONVERTED	See CONVERTED in *Options*.
DBFNR ON	Displays the database id (DBID) and file number (FNR) of the source library in the header line of the source code.
DBFNR OFF	Displays the header line of the source code without the database id (DBID) and file number (FNR) of the source library. This is the default value.
EXPAND	See *expand-option*.
FIELDS	Applies to maps only: displays the field summary; that is, the list of fields in the map.
FIND	Displays only those source lines which contain the specified *value*. See ***Subcommands FIND, REF and SCAN*** for details.
FIND *value*	
FIND ABSOLUTE *value*	
FIND CASESENSITIVE *value*	
FIND ABSC *value*	
FORMAT	Applies to data areas and maps only: displays formatted data area or map, and other items related to the map. See ***Subcommand FORMAT***.
LAYOUT	Applies to maps only: displays the map layout; that is, the map will be displayed as it is displayed to the users at runtime.
NUMBERS ON	Displays the source code with line numbers. This is the default value.
NUMBERS OFF	Displays the source code without line numbers.
PRINT	Prints the listed source.
REF	Displays the line numbers of the source-code lines which contain the specified *value* in a table. See ***Subcommands FIND, REF and SCAN*** for details.
REF *value*	
REF ABSOLUTE *value*	
REF CASESENSITIVE *value*	
REF ABSC *value*	
RULES	Applies to maps only: displays the processing rules used by the map (the rules are displayed in order of fields to which they are assigned, and per field in order of rank).

Subcommand	Function
SCAN	Displays all lines intensified which contain the specified *value*. The source will be scrolled to the first line that contains the *value*. See ***Subcommands FIND, REF and SCAN*** for details.
SCAN *value*	
SCAN ABSOLUTE *value*	
SCAN CASESENSITIVE *value*	
SCAN ABSC *value*	
SCAN= or SC=	Scans for the next occurrence of the last SCAN *value* (or press PF5).
SETTINGS	Applies to maps only: displays the map settings of the map.
ZOOM [*expand-type*...10] *object-name*	Specifying a single *object-name* with the ZOOM command has the same effect as marking the name in the listed source with the cursor (see the section ***Cursor-Sensitive Object Selection***): the selected object will be displayed in a window. If you use asterisk/wildcard notation for the *object-name* or the *object-name-range*, all selected objects will be displayed in a window in the sequence in which they are referenced in the listed source. The specification of an *expand-type* is the same as for the *expand-option*. For an object displayed within a window invoked by ZOOM, the same subcommands (except PRINT, EXPAND and ZOOM) are available as for the normal listed source. Moreover, if you have used asterisk or wildcard notation and several objects are displayed, you can use the commands NEXT and PREV (or PF4 and PF5) to move from one object in the window to the next one or previous one respectively.
ZOOM [*expand-type*...10] *object-name-range*	
.	Exit.

Subcommands FIND, REF and SCAN

Specifying *value*

value is a search string of up to 34 characters to be supplied with the FIND, REF or SCAN command.

If the *value* to be found is identical to the command option CAS, CASESENSITIVE, ABS, ABSC or ABSOLUTE, you have to enclose *value* in single quotation marks (' '), for example: FIND 'ABS'.

If you issue a command without *value*, a window prompts you for the required value. In the prompt window, you can also set options (Y = Yes, N = No) to determine whether the search is to be **absolute** and/or **case-sensitive**.

Absolute Search

If you issue a command with ABSOLUTE, all matches of *value* are found, even if *value* is part of a word. By default, the search is performed on whole words only.

You can also set ABSOLUTE as an option in the prompt window (see *Specifying value*).

Case-Sensitive Search

If you issue a command with CASESENSITIVE, the search is case-sensitive. The string to be found must exactly match the uppercase/ lowercase letter combination specified as *value*. By default, the search is not case-sensitive; all matches are found, regardless of the case in which the letters are specified.

You can also set CASESENSITIVE as an option in the prompt window (see *Specifying value*), and you can change the default setting for CASESENSITIVE in the LIST profile: see SCAN-CASE-SENSITIVE in *Defining an Individual List Profile*.

Absolute Combined with Case-Sensitive

ABSC (or ABSOLUTECASESENSITIVE) is a combination of ABSOLUTE and CASESENSITIVE.

Subcommand FORMAT

This subcommand only applies to stowed data areas (where time stamp of source object and cataloged object are identical) and maps.

For data areas, this subcommand corresponds to the option FORMATTED.

In the List Profile, you can specify how data areas are listed by default:

- formatted (that is, the display resembles a DEFINE DATA statement) or
- unformatted (that is, the display resembles that in the Natural data area editor).

In the List of Objects, you can use the subcommand DEFINE-DATA ON/OFF to set the default for the time the LIST command is being executed.

If data areas are listed formatted by default and if it is not possible to convert the data area source code into DEFINE DATA format, a corresponding message is displayed and the data area is listed unformatted.

When you enter the subcommand FORMAT for a map, a window will be displayed in which you can select one or more additional items related to the map to be displayed:

- Map settings (corresponds to subcommand SETTINGS).
- Map layout (corresponds to subcommand LAYOUT). When you select this item, you have the option to specify filler characters for input fields (AD=A and AD=M) and output fields (AD=O) to make these fields visible. You may specify any character as filler character.
- Field summary (corresponds to subcommand FIELDS).
- Processing rules (corresponds to subcommand RULES).

The items you select are displayed one after the other in the order in which they appear in the selection window.

In FORMAT mode, the same subcommands for scrolling - except B - and the subcommands FIELDS, LAYOUT, PRINT, RULES and SETTINGS are available as for a normal listed source (see above). Additional subcommands are available as described below for each item.

- Additional Subcommands for Map Layout
- Additional Subcommands for Field Summary List
- Additional Subcommands for Processing Rules

Additional Subcommands for Map Layout

S>*n*	Shift map layout *n* columns to the right.
S<*n*	Shift map layout *n* columns to the left.

Additional Subcommands for Field Summary List

EXTEND	Displays the extended field editing information for all map fields. To have the extended field editing information for an individual field displayed, mark the field name on the field summary list with the cursor and press ENTER.
RULES *nn*	Displays the processing rules attached to field *nn* (*nn* being the sequential field number (first column of the field summary list)). To have the processing rules of a field displayed, you can also enter an R in the command line and then mark the field name on the field summary list with the cursor and press ENTER.
SCAN [ABSOLUTE] *value*	Same as for ***Subcommands for Listed Source***.
SCAN =	

Additional Subcommands for Processing Rules

SCAN [ABSOLUTE] *value*	Same as for ***Subcommands for Listed Source***.
SCAN =	

Cursor-Sensitive Object Selection

Within a source that is being listed, you can mark with the cursor the *name* of an object referenced within that source, and the source of the selected object will be listed in a window.

For the source displayed within the window, the same **subcommands** - except PRINT, EXPAND and ZOOM - are available as for the "normal" listed source.

Defining an Individual List Profile

You can define an individual profile for the LIST command. For this purpose, the Natural LIST command provides the subcommand LISTPROF.

This command invokes the profile maintenance tool that

- creates a new LIST profile with default values as used internally, if the LIST profile does not exist;
- allows you to specify general values for single users for the profile parameters in the LIST profile (controlled by Natural Security);
- allows you to specify individual values for single users for the profile parameters in the LIST profile (controlled by Natural Security);
- provides a description of and help information on each profile parameter.

The profile maintenance tool shows a map with the general profile parameters and their current values. Using a PF-key or a command, you can switch to the settings of your own profile data.

Using line commands, parameters can be added to or removed from the user specific profile.

For further information, invoke the help screens of the single fields.

If new parameters are introduced, the profile maintenance tool internal command UPDATE integrates the changes into the LIST profile.

The LIST profile itself is named LIST-PRF and located in library SYSLIB. The default profile that is used for updates of the LIST profile is named LIST-DEF and is also located in library SYSLIB.

LIST Profile Parameters

The table below lists the parameters contained in the LIST profile and the possible values that can be entered. In addition, the table provides a brief description of the parameters. The default parameter values are underlined.

Parameter	Possible Values / Format	Explanation
REUSE-LAST-LIST	N or Y	N The default setting is N (No). Y The last displayed list of objects is reused after execution of commands entered in the **Cmd** column, except for the commands E, ED (Edit), CA (Catalog), UC (Uncat), S, ST (Stow), D, DE (Delete) or RE (Rename).
SCAN-CASE-SENSITIVE	N or Y	N The default setting is N (No). Y Distinguish between uppercase and lowercase letters when scanning the source code for a search value specified with the FIND, REF or SCAN subcommand. See also *Subcommands FIND, REF and SCAN*.
COUNT-SOURCE-LINES	N or Y	N The default setting is N (No). Y Display the number of source lines for source objects in the map with the source directory data (use PF11 to display the source directory data).
SORT-TEXT-MEMBER-NAME	A8	Name of the Natural text member that is used for storing the sorted list. If no name is specified, a generated name is used.
SORT-TEXT-MEMBER-LIBRARY	A8	Name of the library where the Natural text member that is used for storing the sorted list is saved. If no name is specified, the library WORKPLAN is used.
DELETE-SORT-TEXT-MEMBER	N or Y	N The Natural source of type text used for the SORT function is *not* deleted when the LIST command is terminated. Y The Natural source of type text used for the SORT function is to be deleted when the LIST command is terminated. This is the default setting.
PRINT-PAGE-SIZE	N3	Default page size for print. The default setting is 60.
MARK-LONG-LINES	N or Y	Y Long lines in a listed source are marked with L in front of the source line. N Long lines in a listed source are not marked. This is the default setting.

Parameter	Possible Values / Format	Explanation
DATA-AREA-DEFINE-DATA-FORMAT	N or Y	Specify whether data area sources are to be listed in DEFINE DATA format or unformatted. Y A data area source is listed in DEFINE DATA format by default (same as LIST FORMATTED). N A listed data area source is listed unformatted. This is the default setting.
SOURCE-LIST-WITH-DBID-FNR	N or Y	Specify whether the database ID and file number of the source library are to be listed in the header line. N The default setting is N (No). Y The database ID and file number of the source library will be listed in the header line.
TOTAL-LINES-DISPLAY-BATCH	N or Y	In batch mode, specify whether the total number of source lines of all sources from the list of objects is to be displayed at the end of the list of objects. N The total number of source lines is not displayed. This option suggests better performance. Y Display the total number of source lines.
PAGE-TITLE-IN-BATCH	F or A	Specify whether the page title is to be printed on top of every page or on top of first page only. F Print the page title on top of first page only. A Print the page title on top of every page. This is the default setting.
OUTPUT-DESTINATION-BATCH	D, P, W or T	Specify whether the LIST output is to be written in batch mode. D Write to default output device (CMPRINT). P Write to default printer device (CMPRT01). W Write to default work file 1. T Write to a Natural text member.
OUTPUT-TEXT-NAME	A8	Name of the Natural text member that is used for storing the LIST output when OUTPUT-DESTINATION-BATCH=T. If no name is specified, L_*uuuuuu* is used, where *uuuuuu* are the first 6 bytes of the user ID.

Parameter	Possible Values / Format	Explanation
OUTPUT-TEXT-LIBRARY	A8	Name of the Natural library for the Natural text member that is used for storing the LIST output when OUTPUT-DESTINATION-BATCH=T. If no name is specified, the library WORKPLAN is used.
USE-MAINPR	N or Y	Specify whether the MAINPR profile parameter is to be used for the output of the LIST command in batch mode. N Write the output to Report 0. Y Write the output to the printer specified with the MAINPR profile parameter.
SHOW-CAT-TIMESTAMP-VALUES	N or Y	Specify the catalog date and time displayed in an object selection list and in an object directory (LIST DIRECTORY): N Use the date and time values of the cataloged object directory that are adjusted by the values set with the Natural profile parameters DD and TD (see the *Parameter Reference* documentation). This is the default setting. Y Use the time stamp (store clock value) of the cataloged object.
SHOW-CAT-DBID-FNR-IN-LIST-DIR	N or Y	Specify whether the directory of the cataloged object displays the DBID (database ID) and FNR (file number) of the library where the object was cataloged (LIST DIRECTORY): N Do not display the DBID and FNR of the library where the object was cataloged. Y Display the DBID and FNR of the library where the object was cataloged. This is the default setting.
LIST-SHIFT-INCREMENT	N3	Specify the step increment the LIST command uses to scroll right or left in a result list with PF10 or PF11, respectively.

26 LISTDBRM

`LISTDBRM`

Notes:

1. This command is only available with Natural for Db2.
2. `LISTDBRM` has to be issued from the Natural system library `SYSDB2`, which means you have to log on to `SYSDB2` first and then enter the command `LISTDBRM`.

The `LISTDBRM` command is used to display existing DBRMs of Natural programs or Natural programs referencing a given DBRM.

When the command is used, the following menu is displayed:

```
 14:29:18              ***** NATURAL TOOLS FOR SQL *****           2010-02-02
                              - List DBRM -




                     Code Function

                      D   Display DBRMs of Programs
                      R   List Programs Referencing DBRM
                      ?   Help
                      .   Exit

               Code .. _   Library .. _________
                           Member ... _________
                           DBRM ..... _________




Command ===>
  Enter-PF1---PF2---PF3---PF4---PF5---PF6---PF7---PF8---PF9---PF10--PF11--PF12---
        Help        Exit                                                 Canc
```

The following functions are available:

Code	Description
D	Displays programs that have access to Db2, and their corresponding package (DBRM). If no DBRM name is shown, the corresponding program uses dynamic SQL.
R	Lists all programs that use a given package (DBRM). If no DBRM name is specified, all programs that use dynamic SQL are listed.

The following parameters apply:

Parameter	Description
Library	Specifies the name of a Natural library. Library names beginning with SYS are not permitted.This parameter must be specified.
Member	Specifies the name of the Natural program (member) to be displayed. This parameter is optional and can be used to limit the output. If a value is specified followed by an asterisk (*), all members in the specified library with names beginning with this value are listed. If the **Member** field is left blank, or if an asterisk is specified only, all members in the specified library are listed.

Parameter	Description
DBRM	Specifies a valid package (DBRM) name. If left blank, programs that run dynamically are referenced. This parameter applies to function code `R` only.

Sample List DBRM Result Screen

```
14:29:22               ***** LISTDBRM Command *****                2010-02-02


    Library  Name     Type         DBRM       User ID   Date       Time
    -------- -------- -----------  --------   --------  --------   --------
    EXAMPLE  PROG1    Program      PACK1      SAG       2006-03-17 11:10:43
    EXAMPLE  PROG2    Program      PACK1      SAG       2006-03-17 11:10:48
    EXAMPLE  PROG3    Program      PACK2      SAG       2006-03-17 11:11:04
    EXAMPLE  PROG4    Program                 SAG       2006-03-17 11:11:07
```

27 LISTPROF

`LISTPROF`

Notes:

1. This command is only available with Natural for Db2.
2. `LISTPROF` has to be issued from the Natural system library `SYSDB2`, which means you have to log on to `SYSDB2` first and then enter the command `LISTPROF`.

The `LISTPROF` command is used to display existing SQLJ profiles of Natural programs or Natural programs referencing a given SQLJ profile.

When the command is used, the following menu is displayed:

```
 14:29:18              ***** NATURAL TOOLS FOR SQL *****           2023-02-16
                              - List SQLJPROF -




                     Code Function

                      D   Display SQLJ Profiles of Programs                  ↩

                      R   List Programs Referencing SQLJ Profiles            ↩

                      ?   Help
                      .   Exit

               Code .. _   Library .. _________
                           Member ... _________
                           SQLJPROF . _________




Command ===>
  Enter-PF1---PF2---PF3---PF4---PF5---PF6---PF7---PF8---PF9---PF10--PF11--PF12---
        Help        Exit                                                 Canc
```

The following functions are available:

Code	Description
D	Displays programs that have access to Db2 and their corresponding SQLJ profile. If no SQLJ profile name is shown, the corresponding program uses dynamic SQL.
R	Lists all programs that use a given SQLJ profile. If no SQLJ profile name is specified, all programs that use dynamic SQL are listed.

The following parameters apply:

Parameter	Description
Library	Specifies the name of a Natural library. Library names beginning with SYS are not permitted.This parameter must be specified.
Member	Specifies the name of the Natural program (member) to be displayed. This parameter is optional and can be used to limit the output. If a value is specified followed by an asterisk (*), all members in the specified library with names beginning with this value are listed.

Parameter	Description
	If the **Member** field is left blank, or if an asterisk is specified only, all members in the specified library are listed.
SQLJPROF	Specifies a valid SQLJ profile name. If left blank, programs that run dynamically are referenced. This parameter applies to function code `R` only.

Sample List DBRM Result Screen

```
16:16:22              ***** NATURAL TOOLS FOR SQL *****            2023-02-16
                                          - List SQLJPROF -                ↩


 Library  Name      Type        SQLJPROF  User ID      Date       Time
 -------- --------  ---------   --------  --------   --------   --------
 HGK      P017523   Program                HGK      2011-02-23 16:16:25
 HGK      P017622   Program    P017622     HGK      2023-02-02 17:51:32
 HGK      P017622D  Program    P017622     HGK      2022-12-13 16:08:16
 HGK      P017622F  Program    P017622     HGK      2022-12-13 16:08:16
 HGK      P017622R  Program    P017622     HGK      2022-12-13 16:08:16
 HGK      P017622U  Program    P017622     HGK      2022-12-13 16:08:16
 HGK      P017948C  Program                HGK      2011-05-31 10:41:06
 HGK      P017948D  Program                HGK      2011-05-27 16:07:05
 HGK      P017948X  Program                HGK      2011-05-31 09:45:54
```

28 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.

29 LISTSQL

This command is only available with Natural for Db2.



This command generates a list of those Natural statements in the source code of an object which are associated with a database access. Also, it displays the corresponding SQL commands these Natural statements have been translated into. This enables you to view the generated SQLCODE before executing a Natural program which accesses an SQL table.

Syntax Element	Description
object-name <sa>	If you specify a valid object name, the object to be displayed must be stored in the library to which you are currently logged on. If you do not specify an object name or if you specify <sa> (source area), LISTSQL refers to the object currently in the Natural source area. In any case, LISTSQL needs a cataloged or stowed object to perform its functionality.
ALL	If you specify the keyword ALL, the generated SQL statements of one object will be displayed in direct succession; that is, without scrolling. If you omit this keyword, the generated SQL statements contained in the specified object are listed one per page. You can use ALL in online mode and in batch mode. The output format will be the same. The functions **Error** (PF2), **Explain** (PF4) or **Parms** (PF6) are not available. When you specify ALL, you can use a question mark (?) or an asterisk (*) as wildcard character, for example: LISTSQL PGM* ALL. The special characters > and < are allowed, but only at the end of a string; this means, that, for example, ABC<DEF would be an invalid expression.

Sample LISTSQL Screen:

```
 14:50:23               ***** NATURAL TOOLS FOR SQL *****            2009-12-04
 Member DEM2SEL                    - LISTSQL -                   Library SYSDB243
 SQL Builder Version 4.30
 Natural statement  at line 0140                                    Stmt   1 / 1

   SELECT *
       INTO VIEW NAT-DEMO
       FROM NAT-DEMO

 Generated SQL statement   Mode : dynamic  DBRM :                  Line   1 / 3
                                          NDZ  : static       Prof : DEM2PROF     ↩
             Length  68
   SELECT  NAME, ADDRESS, DATEOFBIRTH, SALARY
   FROM    NAT.DEMO
   FOR FETCH ONLY




Command ===>                                          Queryno for EXPLAIN 1____


Enter-PF1---PF2---PF3---PF4---PF5---PF6---PF7---PF8---PF9---PF10--PF11--PF12---
            Error Exit  Expl        Parms                   Prev  Next  Canc
```

If a static DBRM has been generated, the name of this DBRM is displayed in the **DBRM** field of the **LISTSQL** screen; otherwise, the **DBRM** field remains empty.

If a NDZ static generation has been executed, **NDZ : static** is displayed and the SQLJ profile name is displayed after the literal **Prof**. If there has been no NDZ static generation performed, the **NDZ** and **Prof** fields will not appear.

The following screen-specific PF key functions are available:

PF Key (Label):	Function:
PF2 (Error)	This key executes the SQLERR command. If an error occurs during EXPLAIN, you can use this key to get information on Db2 errors.
PF4 (Expl)	With this key, you can execute an EXPLAIN command for the SQL statement currently listed. The query number for the EXPLAIN command (in the field **Queryno for EXPLAIN**) is set to 1 by default, but you can overwrite this default. EXPLAIN is only supported for Natural for Db2.
PF6 (Parms)	You can use this key to display a further screen which lists all parameters from the SQLDA for the currently displayed SQL statement; see sample screen below.
PF10 (Prev), PF11 (Next)	Within the listed results, you can go from one listed SQL statement to another by pressing the corresponding key.

Sample Parameter Screen:

```
 14:55:24               ***** NATURAL TOOLS FOR SQL *****           2009-12-04
 Member DEM2SEL                      LISTSQL                    Library SYSDB243
        NDZ  : static       Prof : DEM2PROF  Sequence : 00008 Sequence2 : 00000  ↩

        Mode : dynamic   DBRM :           Contoken :
                   (3rd/pre)
        static parms : (1st)
                       (2nd)
        SQLDA
                                    DBID : 250  FNR :   1  CMD : S1 0140 08
    Nr  Type     Length     CCSID
     1. CHAR             20       8001 0000 0014 01C4 0000 0000 0800 0000
     2. CHAR            100       8002 0000 0064 01C4 0000 0000 0800 0000
     3. CHAR             10       8003 0000 000A 01C4 0000 0000 0800 0000
     4. DECIMAL         6.2       8004 4000 0602 01E5 0000 0000 0800 0000









Enter-PF1---PF2---PF3---PF4---PF5---PF6---PF7---PF8---PF9---PF10--PF11--PF12---
                  Exit                                                  Canc
```

In static mode, static information is also displayed, which includes the static DBRM name, the Db2 consistency token, and some internal static parameters.

If an NDZ static generation has been executed, **NDZ : static** is displayed. The SQLJ profile name is displayed after the literal **Prof** and the **Sequence** and **Sequence2** fields display the SQLJ sequence number of the statement and the sequence number of the secondary statement if the file server option has been specified for the static generation. If there was no NDZ static generation performed for the program, the **NDZ**, **Prof**, and **Sequence** fields do not appear on the display.

To navigate on the parameters screen, you can use the following PF keys, whose functions are assigned only if the information does not fit on the screen.

PF Key (Label):	Function:
PF6 (top,--), PF9 (bottom,++)	Using these keys, you can go directly to the top (--) or to the bottom (++) of the list.
PF8, PF7 (-)	Using these keys, you can scroll forwards (+) or backwards (-) by pressing the corresponding key.

Using the EXPLAIN Command with Natural for Db2

Important: Before you use the Db2 `EXPLAIN` command, refer to the section *LISTSQL and Explain Functions* in the section *Installing Natural for Db2 on z/OS* in the *Installation for z/OS* documentation.

The EXPLAIN command provides information on the Db2 optimizer's choice of strategy for executing SQL statements. For the EXPLAIN command to be executed, a PLAN_TABLE must exist. The information determined by the Db2 optimizer is written to this table. The corresponding explanation is read from the PLAN_TABLE and displayed via the **EXPLAIN Result** screen.

Sample **Explain Result** Screen:

```
 10:57:47               ***** NATURAL TOOLS FOR SQL *****            2009-12-03
 Queryno 1                       EXPLAIN Result                     Row  1 / 1

                      Estimated cost :   296.6  timerons

Qblock  Plan Mixop Acc. Match Index Pre-  Column- Access-
    No    No   seq type  cols only  fetch fn_eval Creator.Name
------ ----- ----- ---- ----- ----- ----- ------- ----------------------------
     1     1       R               S




       Table-                             Tslock        -- sortn --  -- sortc --
 TabNo Creator.Name                       mode   Method uq jo or gr  uq jo or gr
------ --------------------------------  ------ ------ -- -- -- --  -- -- -- --
     1 NAT.DEMO                          IS            N  N  N  N   N  N  N  N




Enter-PF1---PF2---PF3---PF4---PF5---PF6---PF7---PF8---PF9---PF10--PF11--PF12---
                  Exit  Info        More   -     +                     Canc
```

If an explanation does not fit on one screen, you can scroll backwards and forwards by pressing PF7 (-) or PF8 (+), respectively.

The value in the **Estimated cost** field is taken from SQLERRD (4) in the SQLCA; it is a rough estimate of the required resources.

With PF4 (Info), the additional information that is provided with the EXPLAINB command is displayed.

30 LISTSQLB

`LISTSQLB`

Notes:

1. This command is only available with Natural for Db2.
2. Before you use the `LISTSQLB` command, refer to *LISTSQL and Explain Functions* in the section *Installing Natural for Db2 on z/OS* in the *Installation for z/OS* documentation.

The command `LISTSQLB` can be executed in batch mode or issued online from the Natural `NEXT` prompt.

If executed online, the following screen is invoked:

```
10:54:35              ***** NATURAL Tools for SQL *****            2006-03-17
                                   - LISTSQLB -




                          Code Function

                           X   Explain all SQL Statements
                           .   Exit

                     Code .. _   Member ... ________
                                 Queryno .. 1____




Enter-PF1---PF2---PF3---PF4---PF5---PF6---PF7---PF8---PF9---PF10--PF11--PF12---
                    Exit                                                  Canc
```

By specifying a valid member name, the explanation of SQL statements can be limited to certain member(s); an asterisk (*) can be used for range specification:

- If you specify a unique member name, all SQL statements contained in this member are explained;
- If you specify a value followed by an asterisk, all SQL statements contained in all members with names beginning with the specified value are explained;
- If you specify an asterisk only (or leave the field blank), all SQL statements of all existing SQL members are explained.

A query number must be specified, so that with each issued `EXPLAIN` command, the newly created explanation is added to the appropriate query number. The default query number is `1`.

To issue the `EXPLAIN` command, enter function code `X` and specify a valid member name and query number; all SQL statements contained in the specified member(s) are explained.

If `LISTSQLB` is executed online, the following screen informs you about the processing status of the command and if any errors have occurred.

```
10:55:24              ***** NATURAL Tools for SQL *****             2006-03-17
                                   - LISTSQLB -
    Queryno : 1                              Member  Stmtno Message

    Current Object :
    Library         TEST
    Member          RTTB--IN

    Statistics :
    Members read    1
       with SQL     1
    SQL statements  7

       Member   Message




       RTTB--IN OK

Press Enter to continue
Command ===>
Enter-PF1---PF2---PF3---PF4---PF5---PF6---PF7---PF8---PF9---PF10--PF11--PF12---
```

If executed in batch mode, error messages are written to a data set referred to by DD name `CMPRINT` (logical printer 0).

31 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.

32 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.

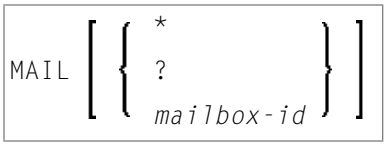
LOGON causes all Natural global data areas and application independent variables (AIVs), all assignments made using the SET KEY statement and retained ISN lists to be released. Data definition modules (DDMs) contained in the DDM buffer area are also released.

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

LOGON *library-id*	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.	
LOGON *library-id password*	The Adabas password; see *Session Parameters* in section *Library Maintenance* of the *Natural Security* documentation.	

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

33 MAIL

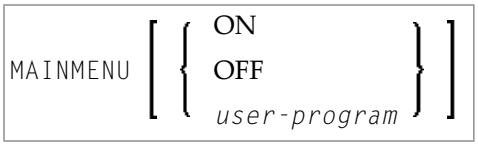


This command is used to invoke a mailbox which is a kind of “notice board” used to broadcast messages under Natural Security. The contents and/or expiration date of the mailbox can be modified.

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

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

34 MAINMENU



This command is used to activate or deactivate the Natural **Main Menu** described in the *Using Natural* documentation.

It is not available via the command line in a remote development environment.

`MAINMENU`	Switches main menu mode on and opens the Natural **Main Menu**.
`MAINMENU ON`	This is the default.
`MAINMENU OFF`	Switches main menu mode off and closes the Natural **Main Menu**.
`MAINMENU` *`user-program`*	Instead of the Natural **Main Menu**, a user-defined program is invoked, which in turn invokes a user-defined menu.

See also Natural profile parameter `MENU`.

35 NATQVS

This command is used to display information about the characteristics of the machine that is currently executing Natural (for example, the CPU ID). This data can be used to check whether it matches the data specified in the product license.

For further information, see *Mainframe Product Licensing*.

36 NOCOPT

NOCOPT

This command is used to display or modify the current settings of the Natural Optimizer Compiler options as they were specified during Natural startup.

For more information on NOCOPT, see *Activating the Optimizer Compiler* in the *Natural Optimizer Compiler* documentation.

37 NOCSHOW

NOCSHOW

This command is used to provide buffer information on the output generated by the PGEN option of the Natural Optimizer Compiler.

For more information on NOCSHOW, see *Optimizer Options*, section *Output of the PGEN Option*, in the *Natural Optimizer Compiler* documentation.

38 NOCSTAT

NOCSTAT

This command is used to provide statistical data on programs suitable for processing by the Natural Optimizer Compiler.

For more information on NOCSTAT, see *NOCSTAT Command* in the *Natural Optimizer Compiler* documentation.

39 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.

40 PROFILER

```
PROFILER
```

This command is used to invoke the Profiler utility. The Profiler utility monitors the internal process flow of a Natural application and analyzes the performance of the application.

For further information, see *Profiler Utility* in the *Utilities* documentation.

41 RDC

```
RDC { ON  }
    { OFF }
```

This command is used in conjunction with the Profiler utility to activate or deactivate the trace recording function of the Natural Data Collector.

For further information, see *Activating and Controlling the Natural Data Collector* in *SYSRDC Utility* in the *Utilities* documentation.

`RDC ON`	Activates the trace recording function of the Natural Data Collector.
`RDC OFF`	Deactivates the trace recording function of the Natural Data Collector.

See also the terminal commands `%<RDC+` and `%<RDC-` in the *Terminal Commands* documentation and profile parameter `RDC` in the *Parameter Reference* documentation.

42 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.

READ *object-name*	The name of the object to be read. If *object-name* 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.
READ *object-name library-id*	The library in which the object to be read is contained. If both *object-name* and *library-id* are specified, Natural will only read the object if it is stored under the specified library ID.

43 RENAME

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

This command is used to give a Natural 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 **Rename Object** window appears where you can specify the same parameters as in the command line.	
old-name	As *old-name* you specify the existing name of the object to be renamed.	
new-name	As *new-name* you specify the name under which the object is to be stored from now on.	
new-type	When you rename an object in source form, you can also change its object type by specifying the corresponding character for *new-type*. The possible values you can specify for *new-type* are:	
	3	Dialog
	4	Class
	5	Processor
	7	Function
	8	Adapter
	9	Resource
	A	Parameter data area
	C	Copycode
	G	Global data area

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

44 RENUMBER

```
RENUMBER [(n)]
```

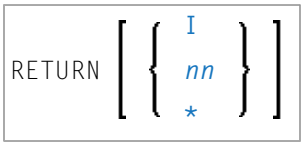
This command is used to renumber the lines in the source code currently in the work area of a Natural editor.

Note: If you want to renumber alphanumeric or Unicode constants, make sure that the RNCONST profile parameter is set to ON.

RENUMBER	If you enter the command without parameter, depending on the total number of source lines in the work area, the following default increment values are used for renumbering:	
	0001 to 0999 lines:	increments of 10
	1000 to 1999 lines:	increments of 5
	2000 to 4999 lines:	increments of 2
	5000 lines and more:	increments of 1
	Starting with line 9999, the line number assigned is always 9999. Be aware that you cannot compile sources that contain more than 9999 lines.	
RENUMBER (*n*)	*n* can be used to specify a value between 1 and 9999 as the increment for renumbering. If the given increment value would cause the renumbering to exceed the 9999 line number limit, the default increment value is used instead.	

See also *Renumbering of Source-Code Line Number References* in the *Programming Guide*.

45 RETURN



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

Application programming interface: USR1026N. See *SYSEXT - Natural Application Programming Interfaces* in the *Utilities* documentation.

RETURN	If RETURN is specified without any parameters, control will be returned to the previous application (as defined with the system command SETUP). All information about this previous application will be deleted. If no previous application exists, control is returned to the initial application. If RETURN is issued and no return point is set, the RETURN command will be ignored. **Under Natural Security:** A LOGOFF command will be executed if RETURN is issued and no return point has been set.
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 *nn*th previous application. When this option is used, all information for applications subsequent to the *nn*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.

46 ROUTINES

ROUTINES

This command is used to ascertain which cataloged objects in the current library use which external subroutines.

All objects in the current library are listed along with the names of the external subroutines they invoke, and the object names of the subroutines in which the external subroutines are contained.

If an object is itself a subroutine, class or function, the long name of the subroutine, class or function it contains is displayed.

47 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 EntireX Broker reason code and associated message. Additionally, the node and server name from the last EntireX 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 RPC (Remote Procedure Call)* documentation.

48 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.

See also:

- *Natural Compiler* in Natural *System Architecture*
- *Object Naming Conventions* in *Using Natural*

RUN	If *program name* 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.
program-name	The name of the program to be run. If *program-name* 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.
library-id	The library in which the program to be run is contained. If both *program-name* and *library-id* 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 *library-id* must not begin with SYS (except SYSTEM).

49 SAVE

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

Related commands: STOW | CATALOG.

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

See also:

- *Object Naming Conventions* in *Using Natural*
- *Natural Compiler* in Natural *System Architecture* for background information

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.

SAVE	If you use the command without *object-name*, the current source object in the source work area will be saved in the current library. An existing source code will be replaced.
SAVE *object-name*	A new source object is created. As *object-name*, 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.
SAVE *object-name library-id*	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 *library-id* after the *object-name*. A new source object is created, if the source object exists, the command is rejected.

50 SCAN

- SCAN Menu Options 192
- SCAN Edit Commands 195
- SCAN via Direct Command 195
- Defining an Individual SCAN Profile 199
- SCAN Under Natural Security 200

SCAN

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

You can scan single objects, all objects with a specific prefix, all objects within a library or all objects of a specific object type. If Predict is installed, you can also scan objects contained in a Predict set.

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.

This chapter covers the following topics:

SCAN Menu Options

When you enter the SCAN command, the SCAN menu will be displayed, providing the following:

Field	Input setting	
Code	T	**Statistics** Returns the following information: ■ the number of objects that were scanned (excluding the skipped objects, see below); ■ the number of objects in which the scan value was found; ■ the number of source-code lines in which the scan value was found. ■ the number of objects that were skipped due to source work area overflow (Natural error NAT0886); increase size of profile parameter ESIZE to scan/replace all required objects. **Note:** Press PF-Key PF5 for SCAN statistics of the last results (if available).
	L	**List of Objects Containing Scan Value** Displays a list of all objects in which the scan setting was found. From the list, you can select individual objects for further processing. If you wish, you can modify the lines directly in the result screens or by using the appropriate SCAN **edit commands** as described below. The scanned lines can be modified for any object, except maps and data areas or locked objects (*Locking of Source Objects*).

Field	Input setting	
		To modify the entire object, enter the E **subcommand** to call the corresponding editor. If any modifications were previously done in the result screen, you are prompted to confirm any updates. Once the object has been edited, the object should be saved and the editor terminated. Scan processing can then continue.
	S	**Object Lines with Scan Value** Displays one after another each source-code line in which the scan value was found. If you wish, you can modify the lines directly in the result screens or by using the appropriate SCAN **edit commands** as described below. The scanned lines can be modified for any object, except maps and data areas or locked objects (*Locking of Source Objects*). To modify the entire object, enter the E **subcommand** to call the corresponding editor. If any modifications were previously done in the result screen, you are prompted to confirm any updates. Once the object has been edited, the object should be saved and the editor terminated. Scan processing can then continue.
Scan value	The string of characters to be scanned for. **Note:** To prevent lower-case characters from being translated to upper-case by Natural, use the terminal command %L.	
Replace value	The value which is to replace the scan value. The **Replace value** option has no effect with maps, data areas, recordings, dialogs or locked objects (*Locking of Source Objects*).	
Library	The ID of the library to be scanned. Default is the current library. If the library specified is SYSTEM, the library in the FUSER file will be scanned. If the name of the specified library begins with "SYS" but is not SYSTEM, the library in the FNAT file will be scanned.	
Object name	The object(s) to be scanned:	
	blank	all objects
	*	
	object-name>	all objects whose names are greater than or equal to *name*
	object-name<	all objects whose names are less than or equal to *name*
	If you wish to scan within 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 LIST.	

Field	Input setting	
	See also *Object Naming Conventions* in the *Using Natural* documentation.	
Selection list	Y	Display a list of objects as specified by Library, Name, Type(s) for Code `T` or `S` (see above). From this list, you can select individual objects (by marking them with any character) from scan processing.
	N	By default, no selection is displayed.
Object type(s)	You can restrict the search to specific object types. One or several types may be specified. For a selection list of possible types, enter a question mark (?) in this field. If you leave this field blank or enter an asterisk (*), objects of any type will be scanned.	
Absolute scan	Y	The scan will be "absolute"; that is, the value to be scanned for will be found in any form, even as part of a longer character string.
	N	By default, the scan is not absolute.
Trace	Y	Activates the trace facility.
	N	By default, the trace facility is deactivated.
Predict set number	Predict set number to be scanned. **Note:** Predict has to be installed. A one- or two-digit number that identifies the retained set to be used for the names of the objects to be processed. A retained set is created with the save set option of the `LIST XREF` command. The value specified for Library is used as Predict set library. If the Predict set number is specified, the value specified for Object name must be an asterisk (*) instead. For detailed information on Predict sets, refer to the *Predict* documentation.	
Predict set user	The ID of the user who created the Predict set. **Note:** Predict has to be installed. If no ID is specified, the value of the system variable `*USER` (see also the *System Variables* documentation) is used.	
Ignore case	Y	Any combination of upper and lower-case letters will be found.
	N	By default, the exact scan value is searched.
Ignore comments	Y	Values found within comments are not displayed or replaced.
	N	By default, all values found are displayed or replaced.

Note: For possible restrictions, see ***SCAN under Natural Security***.

SCAN Edit Commands

The following edit commands can be entered in the command line(s) of the SCAN editor generated by the scan operation:

Command	Function
blank	Continue with normal scan processing.
Q	Terminate scan processing.
.	
EDIT	Edit the object using full-screen editor. **Note:** This command is not available if the Natural program, data area, and map editor are disabled in your environment. For more information, see *NaturalONE as the Default Development Environment* in the *Editors* documentation.
LIST	List the object as it currently appears in the source work area.
LET	Ignore all line changes made after last ENTER.
I	Ignore the object currently being scanned, do not save any modifications, and continue with next object.
.D	Delete line. To indicate that the line has been deleted, the character D will appear next to the line.
.L	Ignore any changes after last ENTER. Will also restore any line previously deleted with the line command .D.

Editing Rules

- The line length of the source object in the result screen is limited to 72 characters. Lines exceeding 72 characters are marked with an L and cannot be modified.
- If the **Replace value** option is used and/or an object is modified in the result screen, the object will always be saved unless an I, Q, or a period (.) is specified before the next object is scanned.
- Lines containing PASSW, PASSWORD=, CIPHER=, or CIPH= will be ignored.

SCAN via Direct Command

The SCAN functions may also be invoked via direct command, in either online mode or batch mode. The SCAN direct command can be specified with **keywords** or with **positional parameters**.

In batch mode, the SCAN command will process only one function per invocation to minimize the repercussions of invalid data being specified.

Caution: If you use scan/replace values that contain lowercase characters or embedded blanks, we strongly recommend that you follow the instructions below to avoid unexpected results.

To use SCAN with lowercase characters or embedded blanks in online mode

1 Enter the command `SCAN` to activate the `SCAN` menu.

2 In the input fields of the `SCAN` menu, enter all required values.

To use SCAN with lowercase characters or embedded blanks in batch mode

1 Specify the command `SCAN` to call the `SCAN` menu.

2 Specify the values in a separate data line and exactly in the order of the map fields (see *Menu Options*), for example:

```
SCAN
S,MOVE LEFT,MOVE RIGHT,SYSTEM,*,N,P,N,N,1,SAG ↩
```

In the example of a batch job above, the value `MOVE LEFT` is replaced by `MOVE RIGHT` for all programs of Predict Set `1` for the user `SAG` in the library `SYSTEM`.

This section covers the following topics:

- SCAN with Keywords
- SCAN with Positional Parameters

SCAN with Keywords

The `SCAN` direct command with keywords consists of the command `SCAN` itself which is followed by keywords.

Keywords can be specified in any order or can be omitted. If a keyword is not used, the default value will be assumed for this keyword.

Caution: Do not mix keywords and **positional parameters** within the `SCAN` direct command to avoid unexpected results.

Syntax of SCAN with Keywords:

```
SCAN keyword=value,...
```

Syntax Explanation:

Keyword	Value	Explanation
FUNC	T, L or S	Function code. For details, see the description of the menu field **Code**.
LIB	max. 8 characters	Name of library to be scanned. For details, see the description of the menu field **Library**. Default value is the current library.
OBJ	max. 9 characters	Object name. The object sources to be scanned. For details, see the description of the menu field **Object name**.
TYPE	max. 14 characters	Object type. The object type(s) to be used for the selection. For details, see the description of the menu field **Object type(s)**.
SVAL	max. 32 characters	Scan value. The character string (***without* embedded blanks**) to be scanned for in each object source. For details, see the description of the menu field **Scan value**.
RVAL	max. 32 characters	Replace value. The character string (***without* embedded blanks**) to be replaced by the scan value. For details, see description of the menu field **Replace value**.
ABSOL	Y or N	Absolute scan. For details, see the description of the menu field **Absolute scan**.
SETNO	max. 2 digits	Predict set number to be scanned. For details, see the description of the menu field **Predict set number**.
SETUSER	max. 8 characters	Predict set user ID. The ID of the user who created the Predict set. For details, see the description of the menu field **Predict set user**.
IGNORE-CASE	Y or N	Search is not case sensitive. For details, see the description of the menu field **Ignore case**.
IGNORE-COMMENTS	Y or N	Search excludes values found in commentary text. For details, see the description of the menu field **Ignore comments**.

Examples of SCAN with Keywords:

```
SCAN FUNC=S,LIB=SYSTEM,OBJ=PGM0*,TYPE=S,SVAL=FETCH ↩

SCAN FUNC=S,SVAL=WRITE,RVAL=PRINT,OBJ=PGM1

SCAN FUNC=S,SETNO=1,TYPE=P,SVAL=GLOBAL
```

SCAN with Positional Parameters

The SCAN direct command with positional parameters consists of the command SCAN itself which is followed by parameters that reference each word on the command line by their position within the command. The order of the positional parameters must be strictly adhered to.

If a positional parameter is not specified, its default value will be assumed. If you want to indicate that a positional parameter is to be omitted, use a comma as shown in *Examples of SCAN with Positional Parameters*.

Caution: Do not mix **keywords** and positional parameters within the SCAN direct command to avoid unexpected results.

Syntax of SCAN with Positional Parameters:

```
SCAN func,scan-value,replace-value,library,object-name,object-type,absolute,set-number,set-user,ignore-case
```

Syntax Explanation:

Positional Parameter	Value	Explanation
func	T, L or S	Function code. For details, see the description of the menu field **Code**.
scan-value	max. 32 characters	Scan value. The character string (***without* embedded blanks**) to be scanned for in each object source. For details, see the description of the menu field **Scan value**.
replace-value	max. 32 characters	Replace value. The character string (***without* embedded blanks**) to be replaced by the scan value. For details, see the description of the menu field **Replace value**.
library	max. 8 characters	Name of library to be scanned. For details, see the description of the menu field **Library**. Default value is the current library.
object-name	max. 9 characters	Object name. The object sources to be scanned. For details, see the description of the menu field **Object name**.
object-type	max. 14 characters	Object type. The object type(s) to be used for the selection. For details, see the description of the menu field **Object type(s)**.
absolute	Y or N	Absolute scan. For details, see the description of the menu field **Absolute scan**.
set-number	max. 2 digits	Predict set number to be scanned. For details, see the description of the menu field **Predict set number**.
set-user	max. 8 characters	Predict set user ID. The ID of the user who created the Predict set. For details, see the description of the menu field **Predict set user**.
ignore-case	Y or N	Search is not case sensitive. For details, see the description of the menu field **Ignore case**.

Positional Parameter	Value	Explanation
ignore-comments	Y or N	Search excludes values found in commentary text. For details, see the description of the menu field **Ignore comments**.

Examples of SCAN with Positional Parameters:

```
SCAN S,ABC,,SYSTEM,PGM0*,P,N

SCAN S,ABC,DEF,SYSTEM,PGM1,*,Y

SCAN T,ABC,,SYSTEM,*,7,N,1,SAG
```

Defining an Individual SCAN Profile

You can define an individual profile for the `SCAN` command. For this purpose, the Natural `SCAN` command provides the subcommand `SCANPROF` or PF-Key `PF10` 'Prof'.

This command invokes the profile maintenance tool that:

- creates a new `SCAN` profile with default values as used internally, if the `SCAN` profile does not exist;
- allows you to specify general values for single users for the profile parameters in the `SCAN` profile (controlled by Natural Security);
- allows you to specify individual values for single users for the profile parameters in the `SCAN` profile (controlled by Natural Security);
- provides a description of and help information on each profile parameter. The profile maintenance tool shows a map with the general profile parameters and their current values.

Using a PF-key or a command, you can switch to the settings of your own profile data.

Using line commands, parameters can be added to or removed from the user specific profile.

For further information, invoke the help screens of the single fields.

If new parameters are introduced, the profile maintenance tool internal command `UPDATE` integrates the changes into the `SCAN` profile. The `SCAN` profile itself is named `SCANPROF` and located in library `SYSLIB`. The default profile that is used for updates of the `SCAN` profile is named `SCANDEFP` and is also located in library `SYSLIB`.

SCAN Profile Parameters

The parameters contained in the SCAN profile correspond to the fields of the SCAN map, see ***Menu Options***.

SCAN Under Natural Security

In a Natural Security environment, you can specify a library security profile to disallow the use of the SCAN system command for a particular library or restrict the use to avoid source modifications. You have the following options:

- Disallow the use of the SCAN system command.
- Allow the use of the SCAN system command, but disallow the system commands LIST and EDIT.
- Disallow the replace function of the SCAN system command. Additionally, disallow the use of the SAVE system command to avoid source modifications through the full-screen editor.

For detailed information on restricting the use of the SCAN system command, see *Command Restrictions* in the *Library Maintenance* section of the *Natural Security* documentation.

If only structured mode is allowed for the library, objects in reporting mode can be scanned, but not modified.

51 SCRATCH

This command is supported for compatibility reasons only. You are strongly recommended to use the `DELETE` command instead.

52 SETUP

- SETUP Syntax Explanation 204
- SETUP/RETURN Example 205

```
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:

Application programming interface: USR1026N. See *SYSEXT - Natural Application Programming Interfaces* in the *Utilities* documentation.

SETUP 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).

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

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 I`.

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

53 SHOWDBS

- Corresponding API USR8211N 209

SHOWDBS

This command is used to view information on the databases available in your environment and the options (like read-only properties) defined for them as shown in the following example:

```
      +-------------------SHOWDBS - Defined Databases-------------------+
      !                                                                 !
      !  DBID DB Type          DB Options                               !
      ! ----- ---------------- ---------------------------------------- !
      !     0 ADABAS V82                                                !
      !    10 ADABAS V82                                                !
      !   111 ADAV7            READ                                     !
      !   147 INCORE DB                                                !
      !   148 PROCESS                                                  !
      !   170 VSAM                                                     !
      !   200 DB2                                                      !
      !   246 DLI                                                      !
      !   247 DLI                                                      !
      !   249 CONNECX                                                  !
      !   250 DB2                                                      !
      ! Default DB type ...... ADABAS V82                              !
      !                                                                !
      +----------------------------------------------------------------+



Top of list

Enter-PF1---PF2---PF3---PF4---PF5---PF6---PF7---PF8---PF9---PF10--PF11--PF12---
                  Exit                          +     ++                Canc
```

The information shown is retrieved from the DB parameter settings (or corresponding NTDB macro) described in the *Parameter Reference* documentation:

DBID	The database ID
DB Type	The type of database: see *Possible Database Types*
Default DB type	The default database type
DB Options	One or more options defined for the database: see *Possible Database Options*

You can page up or down in the **SHOWDBS** window (depending on how many pages exist) and exit with the following PF keys:

PF3	Cancel and close window
PF6	First page
PF7	Previous page
PF8	Next page
PF9	Last page
PF12	Cancel and close window

Corresponding API USR8211N

The application programming interface `USR8211N` in the `SYSEXT` system library provides the same information as the `SHOWDBS` system command.

54 SQLDIAG

SQLDIAG

Note: This command is only available with Natural for Db2.

The SQLDIAG command provides diagnostic information about the last SQL statement (other than a GET DIAGNOSTICS statement) that was executed. This diagnostic information is gathered as the previous SQL statement is executed. Some of the information available through the GET DIAGNOSTICS statement is also available in the SQLCA.

For detailed information about the returned diagnostics information, see the IBM Db2 documentation of the GET DIAGOSTICS statement.

Fields, which are prefixed with a plus sign (+), may contain more data than displayed on the screen. You can display the full contents either when you position the cursor on the field (description or data) and press Enter, or when you enter the abbreviation of the field (which are the capital letters of the description) prefixed by the plus sign (+) in the command line. For example, +SN shows a window with the full value of SERVER_NAME.

The SQLDIAG command can be issued either from the Natural NEXT prompt or from within a Natural program (by using the FETCH statement).

Sample SQLDIAG Diagnostic Information Screen

```
 11:03:12           *** SQLDIAG Diagnostic Information ***           2006-04-15
                           - Statement Information -



 DB2_Last_Row .....................           0
 DB2_Number_Parameter_Markers .....           0
 DB2_Number_Result_Sets ...........           0
 DB2_Return_Status ................           0
 DB2_SQL_Attr_Cursor_Hold .........     _Rowset ..     _Scrollable ...
                                          _Type ..     _Sensitivity ..
 DB2_Number_Rows ..................           0
 Row_Count ........................           0

 More .............................

 Number ...........................           1



Command ===>
Enter-PF1---PF2---PF3---PF4---PF5---PF6---PF7---PF8---PF9---PF10--PF11--PF12---
      Help  Error Exit  Updat                                    Next  Canc
```

```
 11:09:49           *** SQLDIAG Diagnostic Information ***           2006-04-15
                          - Condition Information 1 -

+Server_Name ................. DAEFDB28
+CUrsor_Name .................
 DB2_Error_Code1 .............        -500   DB2_Error_Code2 ...           0
          _Code3 .............           0             _Code4 ...          -1
 DB2_Internal_Error_Pointer ..        -500  +DB2_Sqlerrd1(-6) ..        -500
 DB2_Module_Detecting_Error .. DSNXOTL
+DB2_Ordinal_Token_1 ......... HGK.DEMO
 DB2_Row_Number ..............           0
 DB2_Line_Number .............           0
 DB2_Returned_SQLCode ........        -204
 DB2_Reason_Code .............           0
 Returned_SQLState ...........       42704
 DB2_Message_ID .............. DSN00204E
 Message_Octet_Length ........           0
+Message_Text ................ HGK.DEMO IS AN UNDEFINED NAME


Command ===>
Enter-PF1---PF2---PF3---PF4---PF5---PF6---PF7---PF8---PF9---PF10--PF11--PF12---
      Help  Error Exit  Updat                                Prev  Next  Canc
```

```
11:14:41           *** SQLDIAG Diagnostic Information ***           2006-04-15
                          - Connection Information -



DB2_Authentication_Type ..
DB2_Authentication_ID .... GGS

DB2_Connection_State .....           0
DB2_Connection_Status ....           0
DB2_Encryption_Type ......
DB2_Product_ID ........... DSN08010
DB2_Server_Class_Name .... QDB2 for DB2 UDB for z/OS









Command ===>
Enter-PF1---PF2---PF3---PF4---PF5---PF6---PF7---PF8---PF9---PF10--PF11--PF12---
      Help  Error Exit  Updat                               Prev        Canc
```

55 SQLERR

SQLERR

Note: This command is only available with Natural for Db2.

The SQLERR command is used to obtain diagnostic information about the most recent SQL error.

When an SQL error occurs, Natural issues an appropriate error message. When you enter the SQLERR command, the following information on the most recent SQL error is displayed:

- the Natural error message number;
- the corresponding reason code (if applicable);
- the variables SQLSTATE and SQLCODE returned by Db2;
- the corresponding error message.

The SQLERR command can be issued either from the Natural NEXT prompt or from within a Natural program (by using the FETCH statement).

Sample SQLERR Diagnostic Information Screen (Natural for Db2)

```
                ***** SQLERR Diagnostic Information *****
---------------------- NATURAL SQL Interface Codes -------------------------
Return Code: 3700    Reason Code: 0      SQLSTATE : 52003    SQLCODE: -206
--------------------------------- SQLCA-------------------------------------
SQLERRP (DB2 Sub routine where error occurred)             :  DSNXOGP
SQLERRD (DB2 Internal State)
        RDS Return Code                                    :         700
        DBSS Return Code                                   :           0
        Number of Rows Processed                           :           0
        Estimated Cost                                     :          11.2
        Syntax error on PREPARE or EXECUTE IMMEDIATE       :           0
        Buffer Manager ERROR Code                          :           0
SQLWARN (Warning Flags)
        Data truncated
        Null Values ignored (AVG,SUM,MAX,MIN)              :
        No. of columns greater than no. of host variables  :
        UPDATE/DELETE without WHERE clause                 :
        SQL Statement not valid in DB2                     :
         Adjustment to DATE/TIMESTAMP Variable made         :
DB2 Error Message :
DSNT4081 SQLCODE = -206, ERROR: THE OBJECT TABLE OR VIEW OF THE INSERT,
              DELETE, OR UPDATE STATEMENT IS ALSO IDENTIFIED IN A FROM CLAUSE
```

56 STOW

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

Related commands: SAVE | CATALOG.

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

See also:

- *Natural Compiler* in Natural *System Architecture*
- *Object Naming Conventions* in *Using Natural*

STOW	If you use the command without *object-name*, 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.
STOW *object-name*	Use this command syntax to store a new object (source and generated code) named *object-name* in the current library. If the object exists in either source or cataloged form, the command is rejected.
STOW *object-name library-id*	If both *object-name* and *library-id* 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 module which is not valid, Natural will display an appropriate error message when the STOW command is issued.

57 STRUCT

- Generate Structured Source into Work Area 220
- Display Structure of Source 223
- Print Structure of Source 224
- Write Structure of Source into Work Area 224

STRUCT

This command serves two purposes:

- You can use it to perform structural indentation of the source code of the Natural object currently in the work area of the editor.
- Various display features make the structure of a program clear to you, thus allowing you to detect any structural inconsistencies.

However, since STRUCT processes Natural sources whether or not they can actually be cataloged, a source will not be parsed for syntactical correctness. Although in most cases, STRUCT will deliver nicely structured source lines, there may be source lines which are ambiguous and which will not be structured as expected.

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.

When you enter the system command STRUCT, the STRUCT menu will be displayed. It offers the following functions:

Generate Structured Source into Work Area

With this function, you can have a source program indented so that the indentation of source-code lines reflects the structure of the program.

This function is the same as that of the editor command STRUCT.

Indentation will take the source-code line length into consideration; that is, a line to be indented will not be shifted beyond the right margin; if “correct” indentation would require a line to be shifted beyond the right margin, it will only be moved as far to the right as possible, but not beyond the margin.

With the Generate function, you can specify the following options:

Field	Explanation	
Source Name	In this field, you enter the name of the source you wish to be structurally indented. The specified source will then be read from the system file into the work area and indented. If you specify no source name, the object currently in the work area of the editor will be indented. If the work area is empty, you must specify a source name.	
Shift setting	In this field, you can enter the number of positions (from 1 to 9) by which source-code lines are to be indented. By default, indentation is by 2 positions.	
Align Comments	Y	Each comment line will be indented as far as the statement line above it; except comment lines which begin at the beginning of a line, these will be not be indented.
	N	Comment lines will not be indented.
	L	Comment lines will be aligned left-justified.
Display Messages	Y	A message indicating that the structured program has been generated into the work area and a list of any source-code lines that could not be "correctly" indented (see above) will be displayed.
	N	No such messages will be displayed.
Return to STRUCT	Y	You will be returned to the `STRUCT` menu after the Generate function has been executed.
	N	You will be returned to the screen from where you issued the `STRUCT` command after the Generate function has been executed.



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 execute the Generate function.

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 the function Generate Structured Source has been applied to it:

```
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
```

Display Structure of Source

With this function, you can display the source code of an object along with several items of information which make the structure of the object clear.

With the Display function, you have the following options:

Field	Explanation	
Source Name	In this field, you enter the name of the source you wish to be displayed. The specified source will then be read from the system file and displayed. If you specify no source name, the object currently in the work area of the editor will be displayed. If the work area is empty, you must specify a source name.	
Display Compressed	Y	Source-code lines on the same structural level will not be displayed. Only those lines will be displayed which cause a change in the structure table on the right-hand side of the screen. From the gap in the sequence of line numbers you can tell how many lines are not shown between two given lines displayed.
	N	All source-code lines will be displayed.
Return to STRUCT	Y	You will be returned to the `STRUCT` menu after the Display function has been executed.
	N	You will be returned to the screen from where you issued the `STRUCT` command after the Display function has been executed.

The following information is displayed:

Line Numbers	For every statement which closes a statement block, the source-code line number of the corresponding statement which initiates the statement block will be displayed to the left of the source code.
Structure Table	To the right of the source code, a table is displayed, which contains indicators for open statement blocks. For each open statement block, a single letter is displayed. The different letters refer to different types of statements (for an explanation of the letters, press PF1). Any structural inconsistency in the source code is indicated by a message being displayed in the structure table.

Example of Display with Structure Information:

```
14:17:47 - Structured Source ABC in Library XYZ -                   2003-02-04
  0010      DEFINE DATA LOCAL                                     *0
  0020      1 EMPL VIEW OF EMPLOYEES                              *0
  0030        2 PERSONNEL-ID                                      *0
  0040        2 FULL-NAME                                         *0
  0050          3 FIRST-NAME                                      *0
  0060          3 NAME                                            *0
  0070      1 VEHI VIEW OF VEHICLES                               *0
  0080        2 PERSONNEL-ID                                      *0
  0090        2 MAKE                                              *0
  0100 0010 END-DEFINE                                            *0
  0110      FIND EMPL WITH NAME = 'ADKINSON'                      *F
  0120        IF NO RECORDS FOUND                                 *FJ
  0130          WRITE 'NO RECORD FOUND'                           *FJ
  0140 0120   END-NOREC                                           *FJ
  0150        FIND (1) VEHI WITH PERSONNEL-ID = EMPL.PERSONNEL-I  *FF
  0160          DISPLAY EMPL.PERSONNEL-ID FULL-NAME MAKE          *FF
  0170 0150   END-FIND                                            *FF
  0180 0110 END-FIND                                              *F
  0190      END                                                   *
  PF1=Help, PF2=Menu, PF3=Exit, PF6=Top, PF12=Cancel.
```

The current content of the work area is not affected by the displayed source.

Print Structure of Source

With this function, you can print the source code of an object along with its structural information.

The Print function corresponds to the function Display Structure of Source, only the output is not displayed on the screen but sent to a printer.

With the Print function, you have the same options as with the Display function.

Write Structure of Source into Work Area

With this function, you can read a source from the system file and write it into the editor work area together with its structure information, plus several lines (line numbers 0000) at the beginning of the source, which explain the structure information.

With the Write function, you have the same options as with the function Display Structure of Source, except that you *must* specify a Source Name.

The source and its structure information are written as text into the work area, and can be edited with the system command EDIT.

58 SYSADA

SYSADA

This command is used to invoke the ADACALL utility which is contained in the library SYSADA.

The ADACALL utility enables you to issue Adabas direct calls (native commands) directly to an Adabas database from Natural for z/OS.

The ADACALL utility can be used for learning purposes or for testing/analyzing various problems or scenarios.

For further information, see *ADACALL - Issuing Adabas Direct Calls* in the *Utilities* documentation.

59 SYSAPI

SYSAPI

This command is used to invoke the SYSAPI utility.

This utility is used to locate application programming interfaces (APIs) provided by Natural add-on products such as Entire Output Management (NOM).

For each API, the utility SYSAPI provides one or more example programs that contain a functional description of the API and that can be used to test the effect of the API.

For further information, see *SYSAPI - APIs of Natural Add-on Products* in the *Utilities* documentation.

60 SYSBPM

SYSBPM

This command is used to invoke the SYSBPM utility.

The SYSBPM utility provides statistical information on the current status of the Natural buffer pool including buffer pool cache and on the objects currently in the buffer pool and buffer pool cache.

SYSBPM also offers administration functions.

For further information, see *SYSBPM Utility - Buffer Pool Management* in the *Utilities* documentation.

61 SYSCP

SYSCP

This command is used to invoke the SYSCP utility.

The SYSCP utility can be used to obtain code page information and to check or change the code page assignment of a source.

For further information, see *SYSCP Utility - Code Page Administration* in the *Utilities* documentation.

See also:

- *Unicode and Code Page Support*
- System command CPINFO

62 SYSDB2

SYSDB2

This command is used to invoke Natural Tools for Db2 if Natural for Db2 is installed.

For further information, see *Using Natural Tools for Db2* in the *Natural for Db2* part of the *Database Management System Interfaces* documentation.

63 SYSDDM

SYSDDM

This command is used to invoke the SYSDDM utility which offers functions that are needed to create and maintain Natural data definition modules (DDMs).

For further information, see *SYSDDM Utility* in the *Editors* documentation.

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.

64 SYSEDT

SYSEDT

This command is used to invoke the SYSEDT utility for Editor Buffer Pool Services. The SYSEDT utility is intended for Natural administrators only. It is used to do the following:

- display parameters and runtime information concerning the editor buffer pool,
- modify parameters,
- delete logical work and recovery files.

For further information, see *SYSEDT Utility - Editor Buffer Pool Administration* in the *Utilities* documentation.

65 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 *Utilities* documentation.

66 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 *Utilities* documentation.

67 SYSEXV

SYSEXV

This command is used to invoke the SYSEXV utility.

The SYSEXV utility gives you access to examples of new features available in the current and in some earlier versions of Natural.

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

68 SYSFILE



This command is used to invoke the SYSFILE function of the SYSTP utility. This function provides information on the work files and print files available.

The information provided by the SYSFILE command can also be obtained with the application programming interface USR1007N. See *SYSEXT - Natural Application Programming Interfaces* in the *Utilities* documentation.

SYSFILE	If you enter only the command SYSFILE itself, work file *and* print file assignments are displayed sequentially.
SYSFILE WORKFILE	The work file assignments are displayed separately.
SYSFILE PRINTER	The print file assignments are displayed separately.

For further information, see *Natural Print/Work Files - SYSFILE* in the section *General SYSTP Functions* of the *Utilities* documentation, and the platform-specific information on print file and work file support in the *Operations* documentation.

69 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 - Object Maintenance* in the *Utilities* documentation.

70 SYSNCP

SYSNCP

This command is used to invoke the SYSNCP utility.

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

71 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 *Utilities* documentation.

72 SYSPARM

SYSPARM

This command is used to invoke the SYSPARM utility. You use this utility to create and maintain strings of Natural profile parameters which are stored as Natural profiles.

These Natural profiles can be invoked with the profile parameter PROFILE for the Natural session start.

The SYSPARM command has various parameters mainly used for batch mode (see *Direct Commands and Batch Processing* in the description of the SYSPARM utility).

For further information, see *SYSPARM Utility* in the *Utilities* documentation.

73 SYSPROD

- About SYSPROD 258
- Using the Command in Online Mode 258
- Using the Command with Additional Keywords (Recommended in Batch Mode) 259
- Application Programming Interfaces 259

SYSPROD

About SYSPROD

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

Using the Command in Online Mode

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

- product name
- product version (see also *Version* in the *Glossary*)
- installation date
- installation component: I (INPL) or N (Nucleus)

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 the extended product information.
HI	Display the history of product information.
SC	Display the subcomponents of the product.

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

Using the Command with Additional Keywords (Recommended in Batch Mode)



For batch processing or for an online output without dialog features such as line command input fields, you can issue the command SYSPROD with additional keywords and/or by specifying a product code (for example, NAT) as a parameter.

Keyword/Parameter	Description
ALL	Display a list of all products installed.
<product-code>	Display unformatted information on a specific product.
EX	Display the extended product information.
HI	Display the history of the product information.
SC	Display the subcomponents of the product.

Note: If, in batch mode, the system command SYSPROD is issued without any keyword, the same maps will be used as in online mode; however, the online input must be simulated in this case, for example, by using the terminal command %K*nn*. This method is susceptible to produce errors, and its use is therefore not recommended in batch mode.

Application Programming Interfaces

The following application programming interfaces enable you to get SYSPROD information. They are available in the library SYSEXT:

- USR0050N
- USR2031N

See *SYSEXT Utility - Natural Application Programming Interfaces* in the *Utilities* documentation.

74 SYSPROF

SYSPROF

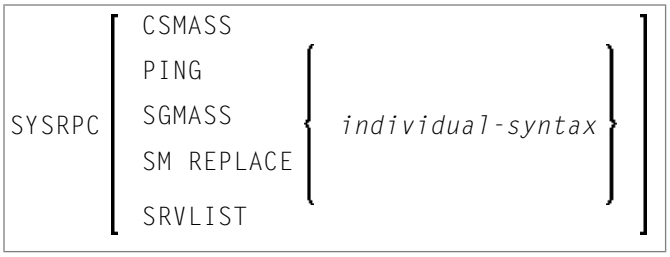
This command is used to display the current definitions of the Natural system files.

For each system file, the following information is displayed:

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

Application programming interfaces: USR0010N, USR2013N, USR3013N. See *SYSEXT - Natural Application Programming Interfaces* in the *Utilities* documentation.

75 SYSRPC



This system command invokes the `SYSRPC` utility which is used to maintain remote procedure calls.

You can specify direct commands with the `SYSRPC` system command in order to perform RPC-specific tasks. These tasks and the *individual-syntax* that applies to the direct commands are described in detail in *SYSRPC Utility* in the *Utilities* documentation:

Direct Command	Purpose and Related Topics
`CSMASS`	Calculates buffer sizes required for RPC calls. See *Calculating Size Requirements* and *Using the SYSRPC CSMASS Command*.
`PING`	Pings a single or all defined servers. See *Pinging an RPC Server* and *Using the SYSRPC PING Direct Command*.
`SGMASS`	Generates multiple interface objects. See *Generating Multiple Interface Objects* and *Using the SYSRPC SGMASS Command*.
`SM REPLACE`	Replaces single or multiple items in a service directory See *Replacing Items in the Service Directory*.
`SRVLIST`	Provides information on Natural RPC servers registered on EntireX Broker: see *Listing Servers Registered on EntireX Broker*.

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 RPC (Remote Procedure Call)* documentation.

76 SYSTP

SYSTP

This command is used to invoke the SYSTP utility which allows you to monitor and control various TP-monitor-specific characteristics of Natural.

For further information, see *SYSTP Utility* in the *Utilities* documentation.

77 TECH

TECH

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

- user ID
- library ID
- Natural version (see also *Version* in the *Glossary*)
- startup transaction
- Natural Security indicator
- Natural editors: Indicates whether the Natural program, data area, and map editors are disabled or enabled (see *EDITOR system variable)
- operating system name and version
- machine class
- hardware
- IBM architecture level supported on the current IBM processor (z/OS)

  0 (zero) denotes that architecture levels are not supported.
- TP monitor (z/OS and Windows (*TPSYS) in remote configuration only)
- device type
- terminal ID (z/OS 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 Natural object and all objects on higher levels, as well as the line numbers of the statements invoking the subordinate objects (z/OS and Linux 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.

Application programming interface: `USR2026N`. See *SYSEXT - Natural Application Programming Interfaces* in the *Utilities* documentation.

78 TEST



This command is used to invoke the debugger.

TEST	If you enter the system command TEST without parameters, the main menu of the debugger is displayed.
TEST ON	Activates the test mode of the debugger.
TEST OFF	Deactivates the test mode of the debugger.
debugger-commands	The direct commands that execute debug functions are explained in the section *Command Summary and Syntax* in the *Debugger* documentation.

To invoke the debugger from any point in an application, you can also use the terminal command %<TEST.

For further information, see the *Debugger* documentation.

The *Utilities* documentation also contains information on other Natural utilities used for online testing and monitoring.

Note Concerning Natural Single Point of Development:

This command is for Natural for z/OS only. If a user has written a program called TEST, then Natural will execute that program when this command is issued in a Windows or Linux local environment. If there is an active connection to a development server on z/OS, the Natural for z/OS utility TEST is invoked when this command is issued under Natural for Windows.

79 TEST DBLOG

```
TEST DBLOG [parameters]
```

This command invokes the DBLOG utility, which is used for logging database calls.

TEST DBLOG	Activates or deactivates the DBLOG utility.
parameters	The parameters that apply to TEST DBLOG are explained in the section *TEST DBLOG Command* of the *Utilities* documentation.

For further information, see *DBLOG Utility - Logging Database Calls* in the *Utilities* documentation.

The *Utilities* documentation also contains information on other Natural utilities used for online testing and monitoring.

Note Concerning Natural Single Point of Development:

If there is an active connection to a development server on z/OS, the Natural utility DBLOG is invoked when this command is issued under Natural for Windows.

80 UNCATALOG

This command is supported for compatibility reasons only. You are strongly recommended to use the `DELETE` command instead.

81 UNLOCK

- Unlocking Natural Objects 276
- UNLOCK Parameter Descriptions 277
- Parameter Processing and Display of Objects Found 278
- Batch Processing 279

This command is used for local unlocking of Natural source objects in a Natural for z/OS environment.

It is used to view source objects that are locked and to unlock them if required. This command is recommended for use by the Natural administrator only. However, the administrator can enable the use of this command for each user profile in Natural Security.

Notes:

1. As a prerequisite for using the `UNLOCK` system command, the profile parameter `SLOCK` must be set to `PRE`.
2. If the number of locked records found is high, it may happen that the list displayed is not sorted. Remedial action: Increase the size of the work buffer used by the sort program; see keyword subparameter `WRKSIZE` of profile parameter `SORT`.

This chapter covers the following topics:

For further information, refer to *Locking of Source Objects* in the *Editors* documentation and profile parameter `SLOCK` in the *Parameter Reference*.

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

Unlocking Natural Objects

If the system command `UNLOCK` is used without parameters, a screen appears where you can enter the parameters.

```
UNLOCK
```

The following shows the direct command syntax for unlocking Natural objects.

```
UNLOCK [NATURAL] [OBJECT] object-name
       [TYPE object-type]
       [LIBRARY library-name]
       [DBID dbid] [FNR fnr]
       [PASSWORD password] [CIPHER cipher]
       [USER locked-by]
       [DATE locked-on [locked-on2]]
```

UNLOCK Parameter Descriptions

The object name must be defined in each case. If any of the other parameters is not specified, the corresponding default value will be used.

Parameter	Format/Length	Default Value	Description
object-name	A33	*	The name of the object to be unlocked. Asterisk notation (*) or ">" can be used.
object-type	A1	*	**Natural object types:** In place of *object-type*, you may specify one of the object type codes shown below or an asterisk (*). ■ P - Program ■ 4 - Class ■ N- Subprogram ■ S - Subroutine ■ 7 - Function ■ 8 - Adapter ■ C - Copycode ■ H - Helproutine ■ T - Text ■ M - Map ■ L - Local Data Area ■ G - Global Data Area ■ A - Parameter Data Area ■ V - DDM (View)
library-name	A8	*	The name of the library where the locked object is in. Asterisk notation (*) can be used.
dbid	A5	current database ID	The database ID of the defined library. Specify asterisk (*) or in format N5. When asterisk notation (*) is used, only the current FNAT, FUSER and FDIC system files are scanned.
fnr	A5	current file number	The file number of the defined library. Specify asterisk (*) or in format N5. When asterisk notation (*) is used, only the current FNAT, FUSER and FDIC system files are scanned.

Parameter	Format/Length	Default Value	Description
password	A8	blank	If used, the password for the specified system file (*dbid* and *fnr*). Needs not to be specified, when the *dbid* and *fnr* of the current FNAT or FUSER is used.
cipher	A8	blank	If used, the cipher key for the specified system file (*dbid* and *fnr*). Needs not to be specified, when the *dbid* and *fnr* of the current FNAT or FUSER is used.
locked-by	A8	current user ID or asterisk (*) with batch mode and security flag "F"	The ID of the user who caused the object to be locked. Asterisk notation (*) can be used. If Natural Security is used, it can be changed only if the security unlock flag is set to "F" (forced unlock) in the Natural Security user profile.
locked-on	A10	blank	The two date parameters are available to provide for the different date formats: 2005-09-28 (date format according to the DTFORM profile parameter) 2005-09-28 11:27:20 Today Today + *nnnn* Today - *nnnn* Yesterday
locked-on2	A8		

Note: Locking can also be enabled locally on a Natural for z/OS server. In this case, the following limitations apply: The *application-name* cannot be used as a selection criterion. For *dbid* and *fnr*, the current FNAT and FUSER system files are searched if asterisk notation (*) is used.

Parameter Processing and Display of Objects Found

If the parameter(s) specified is (are) valid and a complete object name is specified and if the corresponding object is found and it was locked by the current user, this object is unlocked immediately and a corresponding message is displayed. This applies under the condition that the object name is specified directly without using asterisk notation (*) and the current user tries to unlock his own locked records.

If any of the parameters specified is invalid or if no objects are found, the unlock screen with an error message will appear.

In the following cases, the locked objects found are listed on the screen where they can be unlocked using the line command U (see below):

- if you used asterisk notation (*) or ">" (where applicable),
- if you did not specify a specific object name.

Note: When asterisk notation (*) is used for object type and library, the locked DDMs have also to be listed by scanning the current FDIC system file.

Unlock List

Function Keys

The unlock list provides the following function keys:

PF1	Help	Invoke help.
PF3	Exit	Return to unlock list.
PF6	--	Top of list.
PF7	-	Page backward.
PF8	+	Page forward.
PF9	++	End of list.
PF10	<	First part of information (type, library, database ID, file number).
PF11	>	Second part of information (locked by, locked on).
PF12	Cancel	Cancels the UNLOCK command.

Line Command

U	In the **Cmd** column of the unlock list, you can enter the command U in a single line or in multiple lines to unlock the corresponding object(s). Successful unlocking is indicated by an "unlocked" message in the **Message** column.

Batch Processing

If no error occurred, all locked objects found are unlocked and a corresponding message appears.

82 UPDATE

```
UPDATE { ON  }
       { OFF }
```

This command is used to control database updates performed by a Natural program.

`UPDATE ON`	Database updates are allowed.
`UPDATE OFF`	Database updates are not allowed. A database update is not performed when a program with an `UPDATE`, `STORE`, or `DELETE` statement executes. Instead, a NAT1010 warning message is issued during the next screen I/O operation. In addition, a database loop that contains an `UPDATE` or `DELETE` statement does not place the records in hold status (no `read with hold`).

83 WHICH

- Direct Command Syntax 284
- WHICH Parameter Description 284
- Default Output of WHICH Command 284
- Full List of WHICH Command 285
- Batch Processing 286
- WHICH Command under Natural Security 286

```
WHICH [ object-name [FULL] ]
      [          ?         ]
```

This command lets you find objects in the libraries used at run time in the Library Search Order (LSO).

Direct Command Syntax

`WHICH`	Displays a window where you can enter an object name with the additional option to select the full output list.
`WHICH object-name`	Displays the object type, the name of the first library in which the object is found, and the DBID and FNR of this library. See *Default Output of WHICH Command*.
`WHICH object-name FULL`	Lists all libraries of the library search order. For each listed library, it indicates whether the cataloged or source object is found. If you press PF5 in the output screen, you can start a new search. See *Full List of WHICH Command*.
`WHICH ?`	Displays the help text of the `WHICH` command.

WHICH Parameter Description

Parameter	Format/Length	Default Value	Description
object-name	A8	none	The name of the object you want to search.

Default Output of WHICH Command

If you enter the `WHICH` command without the `FULL` option, the output looks like the following example:

```
Cataloged subprogram USR1002N found in library SYSEXT(12,124).
```

Full List of WHICH Command

If you enter the `WHICH` command with the `FULL` option, all libraries of the library search order are listed. The output looks like the following example:

```
12:00:49               ***** NATURAL WHICH COMMAND *****             2023-05-25
User XXXX                 - Search object USR1002N -           Library MYLIB

Library   DBID   FNR Type Save/Catalog Date   Message
-------- ----- ----- ---- ------------------ --------------------------------
MYLIB       12   123
MYLIB01     12   123
MYLIB02     12   123
PROTECT                                     Library access denied
SYSEXT      12   124  N   2022-04-20 07:59:36 Cataloged object found
                      N   2011-05-06 16:37:46 Saved object found
SYSTEM      12   123
SYSTEM      12   124









Enter-PF1---PF2---PF3---PF4---PF5---PF6---PF7---PF8---PF9---PF10--PF11--PF12---
      Help        Exit        New   --    -     +     ++                Canc
```

Explanation of Column Headings

Column	Explanation
Library	The names of libraries in the library search order.
DBID FNR	The database ID and file number of the Natural system file containing the library.
Type	The Natural object type code of the object. See Object Type Code in the *Glossary*.
Save/Catalog Date	The save or catalog date of the object.
Message	The result of the search. An empty line indicates that the object was not found in that library.

Function Keys

The output screen of the `WHICH` command with `FULL` option provides the following function keys:

PF Key	Name	Function
PF1	Help	Invoke help.
PF3	Exit	Exit from WHICH command.
PF5	New	New search.
PF6	--	Top of list.
PF7	-	Page backward.
PF8	+	Page forward.
PF9	++	End of list.
PF12	Canc	Cancel the WHICH command.

Batch Processing

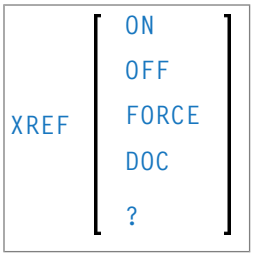
The output of WHICH without FULL option is the same as the online output.

If you call the WHICH command with the FULL option, all resulting libraries are printed in one list.

WHICH Command under Natural Security

For information on WHICH under Natural Security, see *Command Restrictions* in the *Natural Security* documentation.

84 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, helproutines, 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:

`XREF`	If you enter the `XREF` command without parameters, a menu/dialog is displayed where you specify the desired option.
`XREF ON`	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.
`XREF OFF`	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.

XREF FORCE	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.
XREF DOC	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.
XREF ?	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.

85 ZIIP

- Syntax of ziip-subfunction-and-option 290
- ZIIP Syntax Element Description 291
- zIIP Reports 293
- ZIIP Menu for Report Output Options 297
- User Exit for ZIIP PRINT 298
- Application Programming Interface (API) for ZIIP Functions 298

```
ZIIP [ ziip-subfunction-and-option ]
     [ MENU                        ]
```

This system command applies to z/OS batch, batch servers, Com-plete, CICS, TSO and IMS TM environments that support the use of the IBM System z Integrated Information Processor (zIIP). Prerequisite: The required Natural for zIIP product(s) must be installed at your site.

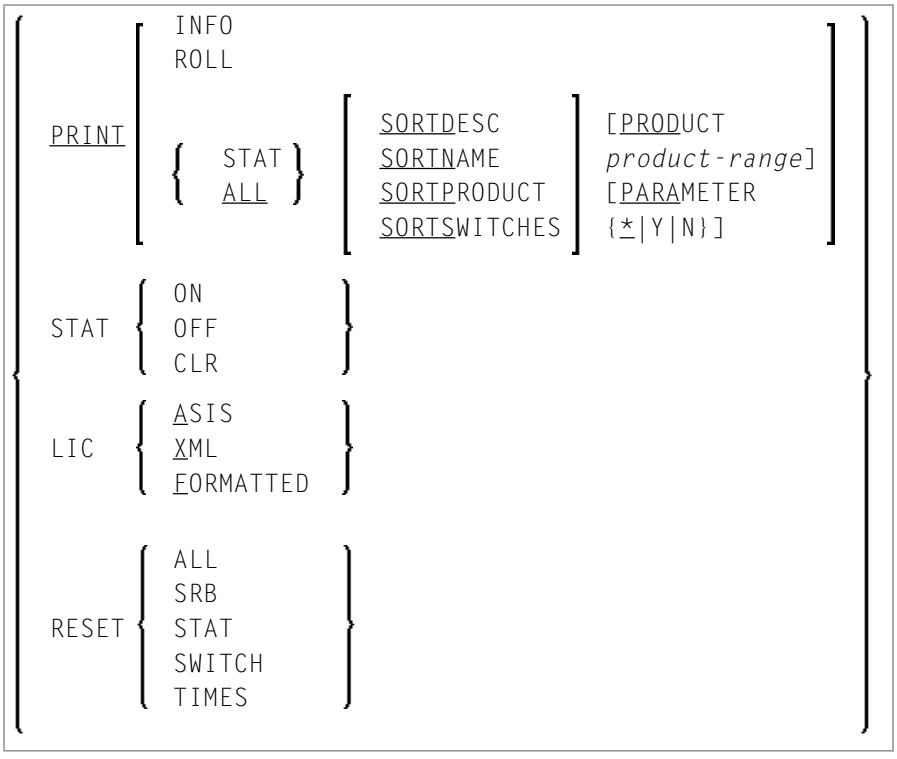
If this system command is issued from a different environment and/or without zIIP support enabled, and/or if the required Natural for zIIP product is not installed, appropriate error messages occur.

The `ZIIP` system command provides status and processing information on the zIIPs used in your environment and the license file required for the Natural for zIIP product installed at your site. In addition, you can obtain statistics on the Natural components that cause switches to the general central processor (GCP) and manage the component switch counting.

For detailed information on zIIP support, usage and monitoring, refer to the *Natural for zIIP* documentation.

This chapter covers the following topics:

Syntax of ziip-subfunction-and-option



ZIIP Syntax Element Description

Subfunction	Option	Explanation
PRINT*	INFO	Displays the **zIIP Processing Information** report.
	ROLL	Displays the **Roll Server zIIP Information** report.
	STAT	Displays the **zIIP Component Switch Statistics** report if component switch counting has been activated (see STAT below). See also *Sort Options for the zIIP Component Switch Statistics* to determine the sort sequence.
	ALL (default)	Displays the **zIIP Processing Information** report first, followed by the **Roll Server zIIP Information** report and the **zIIP Component Switch Statistics** (if activated). See also *Sort Options for the zIIP Component Switch Statistics* to determine the sort sequence.
STAT*	ON	Activates the component switch counting.
	OFF	Deactivates the component switch counting.
	CLR	Clears the component switch table.
LIC	ASIS	Displays the XML-formatted zIIP license file left-justified.
	XML	Displays the XML-formatted zIIP license file.
	FORMATTED (default)	Displays the zIIP license file in free format without brackets.
RESET	ALL	Reset all counters
	SRB	Reset the start SRB counter
	STAT	Reset the component switch table (equivalent to STAT CLR)
	SWITCH	Reset the TCB switch counter
	TIMES	Reset all CPU time values

* Alternatively, you can set the keyword subparameters STAT and PRINT of the profile parameter ZIIP and the macro NTZIIP respectively. See *Keyword Subparameters* in the section *ZIIP - zIIP Processing (z/OS only)* in the *Parameter Reference* documentation.

The RESET function does not affect the roll server zIIP information counters and CPU times.

Sort Options for zIIP Component Switch Statistics

Keyword	Sort Order
SORTDESC	The report is sorted ascending by the component description.
SORTNAME	The report is sorted ascending by the component name.
SORTPRODUCT	The report is sorted ascending by the product name.
SORTSWITCHES (default)	The report is sorted descending by the number of TCB switches (default).

Selection Criteria for zIIP Component Switch Statistics

Keyword	Description	
PRODUCT	Lists the components of a single product or a range of products specified with *product-range*. Valid values for *product-range* are:	
	ppp	A valid product code. Select the product that corresponds to the three-character product code *ppp* as shown in the **Prd** column of the **zIIP Component Switch Statistics** report.
	*p** or *pp**	One or two leading characters. Selects all products whose product codes start with *p** or *pp**.
	ppp>	A start value. Selects all products with product codes greater than or equal to *ppp*.
	ppp<	An end value. Selects all products with product codes less than or equal to *ppp*.
PARAMETER	Specifies the components to be listed. Valid values are:	
	*	Selects all components (default).
	Y	Selects only components with a P indicator denoting that component switches can be influenced by a parameter.
	N	Selects only components (without a P indicator) whose component switches cannot be influenced by a parameter.

zIIP Reports

This section describes the reports and statistical data provided by the `ZIIP` system command.

- zIIP Processing Information
- Roll Server zIIP Information
- zIIP Component Switch Statistics

zIIP Processing Information

The **zIIP Processing Information** report looks similar to the example below:

```
+---------------- zIIP Processing Information ---------------+
|                                                            |
|            Advanced zIIP Support Enabled                   |
|                                                            |
|  General central processors (GPs)             6            |
|  Integrated Information Processors (zIIPs)    1            |
|   zIIP SMT threads                            2            |
|  zIIP normalization factor                10.98            |
|                                                            |
|  Switches into TCB mode                 1590303            |
|  SRB starts                                  80            |
|                                                            |
|  Total enclave CPU time            01:00:56.411  (100.00%) |
|   Enclave GP time                  00:12:38.305   (20.74%) |
|   Enclave zIIP time                00:48:18.106   (79.26%) |
|                                                            |
|  Eligible zIIP CPU time            00:48:22.619  (100.00%) |
|   Enclave zIIP time                00:48:18.106   (99.84%) |
|   Eligible zIIP time on GP         00:00:04.513    (0.16%) |
|                                                            |
|  (zIIP times normalized)                                   |
|                                                            |
|                                                            |
+------------------------------------------------------------+
```

If the z/OS system parameter PROJECTCPU=YES is set and no zIIPs are configured to the current logical partition, the report looks similar to the example below:

```
+--------------- zIIP Processing Information ---------------+
!                                                           !
!           Advanced zIIP Support Enabled (PROJECTCPU)      !
!                                                           !
! General central processors (GPs)             2            !
! Integrated Information Processors (zIIPs)    0            !
!                                                           !
! zIIP normalization factor                 0.00            !
!                                                           !
! Switches into TCB mode                     938            !
! SRB starts                                   6            !
!                                                           !
! Total enclave CPU time            00:00:06.091  (100.00%) !
!  Enclave TCB time                 00:00:04.640   (76.19%) !
!  Enclave SRB time (*)             00:00:01.452   (23.81%) !
!                                                           !
! (*) possible zIIP offload                                 !
!                                                           !
!                                                           !
!                                                           !
+-----------------------------------------------------------+
```

This report provides the following status and processing information:

Report Item	Explanation
Advanced zIIP Support Enabled	Indicates that the current Natural session runs with full zIIP enablement.
General central processors (GPs)	The number of GPs running under your z/OS system.
Integrated Information Processors (zIIPs)	The number of zIIPs running under your z/OS system.
zIIP SMT threads	The number of logical zIIPs running under your z/OS system if SMT (Simultaneous Multi-Threading) is active.
zIIP normalization factor	The zIIP normalization factor indicates the ratio of zIIP to GP speed. This factor tells you how fast your zIIP runs compared to a throttled GP with reduced power. In the **example** above, the value of `10.98` means that one zIIP is about eleven times faster than one GP.
Switches into TCB mode	The number of switches from SRB (service request block) into TCB (task control block) processing mode due to any system service calls that had to be issued. The corresponding switches back into SRB mode are not counted.
SRB starts	The number of times an SRB process was started. The counter is incremented by 1 for the first SRB at each session start and for the SRB restart after an abnormal termination in SRB mode.
Total enclave CPU time	The total CPU time (GP plus zIIP) consumed by the Natural WLM enclave.
Enclave GP time	The total GP time consumed within the Natural WLM enclave.
Enclave zIIP time	The total zIIP time consumed within the Natural WLM enclave.

Report Item	Explanation
Eligible zIIP CPU time	The total CPU time qualified for zIIP.
Eligible zIIP time on GP	The CPU time within the Natural WLM enclave on the GP qualified for zIIP but not used by it. A non-zero value means that zIIP-eligible workload could not be offloaded because no zIIP was available.

For details on the statistics items, refer to *Natural zIIP Processing: TCBs, SRBs and Enclaves* and *Calculating the CPU Time* in the *Natural for zIIP* documentation.

For hints on improving zIIP exploitation, refer to *Tuning zIIP Usage* in the *Natural for zIIP* documentation.

PF Keys Available on the zIIP Processing Information Screen

ENTER or PF5	Updates the report data.
PF7	Switches to the **zIIP Component Switch Statistics**. (Applies to `PRINT ALL` only.)
PF8	Switches to the **Roll Server zIIP Information** report. (Applies to `PRINT ALL` only.)

Roll Server zIIP Information

The **Roll Server zIIP Information** report looks similar to the example below:

```
+------------- Roll Server zIIP Information -------------+
!                                                        !
! Roll Server zIIP CPU Time Usage in Subsystem QA82      !
!                                    Job Name QAROLL82   !
!                                                        !
! Total WLM enclave CPU time ....                 387524 !
! Qualified zIIP CPU time .......                 387252 !
! Eligible zIIP CPU time on GCP                      133 !
! Total WLM enclave zIIP CPU time                 387119 !
! Total WLM enclave zIIP CPU time (%)              99.89 !
!                                                        !
! (All times in milliseconds; zIIP times normalized.)    !
!                                                        !
+--------------------------------------------------------+
```

This report provides the following status and processing information:

Report Item	Explanation
Total WLM enclave CPU time	The total CPU (GCP plus zIIP) time consumed by the current WLM enclave.
Qualified zIIP CPU time	The time qualified for zIIP usage on SRB, which is the sum of the eligible plus the total zIIP CPU time.
Eligible zIIP CPU time on GCP	The CPU time on the GCP qualified for a zIIP but not used by it. A non-zero value means that zIIP-eligible workload could not be offloaded because no zIIP was available.
Total WLM enclave zIIP CPU time	The total zIIP CPU time consumed by the current WLM enclave.
Total WLM enclave zIIP CPU time (%)	The percentage of the total zIIP CPU time consumed by the current WLM enclave. In the **example** above, Natural was able to offload 99.88 % of the total CPU load to a zIIP.

zIIP Component Switch Statistics

The **zIIP Component Switch Statistics** report lists all non-zero TCB switches and the components that cause these switches as shown in the following example:

```
+------------------ zIIP Component Switch Statistics -------------------------+
!                                                                              !
! Prd Component Description                              TCB Switches   %   P !
! ---------------------------------------------------------------------------- !
! NAT CMDBAS   Call Adabas ..............................     321587 37.43 P !
! NSB NSBCNXP  DB2 ConnecX access .......................     140356 16.33   !
! NVS NVSEXSH  VSAM I/O scheduler .......................     120904 14.07   !
! RPC RPC2ETB  RPC interface to EntireX Broker Stub .....     108488 12.62 P !
! NAT NATGWSTG Request handler for Natural Development Ser      94914 11.04 P !
! NAT CMPINIT  Session initialization exit ..............      58361  6.79   !
! ADA LNKDWA_Z TP dependent wait ........................      10947  1.27   !
! NAT CMWAIT   Wait for specified interval ..............       1234  0.14 P !
! COR COR00009 Program delete ...........................        612  0.07   !
! NDB NDBWLI   DB2 IFI calls (DSNWLI) and RRSAF (DSNRLI) a        527  0.06 P !
! NAT CMFREMP  Release working storage ..................        373  0.04 P !
! COR COR00001 Lock contention ..........................        261  0.03   !
! NAT CMDLET   Release external module ..................        213  0.02 P !
! NAT ZPINIT   Initialization of zIIP support ...........        193  0.02   !
! NAT CMABEND  Controlled abend routine .................        133  0.01   !
! ---------------------------------------------------------------------------- !
!     Total                                                 859103         !
!------------------------------------------------------------------------------+
```

Components are service routines of the Natural nucleus, Natural add-on products or other products (for example, Adabas) that cause TCB switches when executing, for example, Adabas calls or external `CALLNAT` requests. The report contains the following:

- The code of the product (for example, `NAT` for Natural) to which the component belongs,

- The name and description of the component,
- The number of switches per component,
- The percentage of the component switches,
- The indicator P set for component switches that can be influenced by a **parameter**, and
- The total number of component switches.

PF Keys Available on the zIIP Component Switch Statistics Screen

ENTER or PF5	Updates the statistics data.
PF4	Opens a window where you can specify selection criteria (product range and component switches that can be influenced by a parameter) for the **zIIP Component Switch Statistics** report.
PF7	Scrolls backward one page (in a long list) or switches to the **zIIP Processing Information** report. (Only applies on the first page of a list and if PRINT ALL is used.)
PF8	Scrolls forward one page (in a long list) or switches to the **zIIP Processing Information** report. (Only applies on the last page of a list and if PRINT ALL is used.)
PF9	Toggles between the two displays described for SORTPRODUCT and SORTSWITCHES in *Sort Options for zIIP Component Switch Statistics*.
PF10	Toggles between the two displays described for SORTDESC and SORTNAME in *Sort Options for zIIP Component Switch Statistics*.
PF11	Toggles between the two displays described for SORTDESC and SORTSWITCHES in *Sort Options for zIIP Component Switch Statistics*.

ZIIP Menu for Report Output Options

In online mode, you can use a menu to set the ZIIP report options according to your requirements.

To display the ZIIP menu

- At any command prompt, enter the following:

```
ZIIP MENU
```

The **Main Menu** of the ZIIP command is displayed, in which you can select the required subfunction and option(s) described in *Syntax Element Description*.

User Exit for ZIIP PRINT

You can use the user exit routine ZIIPUX01 to specify the default settings for the PRINT options of the ZIIP system command. The source is provided as subprogram ZIIPUS01 in the Natural system library SYSEXT. Copy the cataloged object of ZIIPUX01 to the Natural system library SYSLIB, SYSLIBS or SYSTEM. For details, see the subprogram ZIIPUS01 in the system library SYSEXT.

Application Programming Interface (API) for ZIIP Functions

You can use the Natural application programming interface USR8204N to perform all functions of the ZIIP system command. For details, see the text object USR8204T and the program USR8204P in the Natural system library SYSEXT.

Index

A

AIV
system command, 19

B

BUS
system command, 21

C

CATALL
system command, 23
CATALOG
system command, 31
CHECK
system command, 33
CLEAR
system command, 35
COMPOPT
system command, 37

D

DELETE
system command, 63
DUMP
system command, 67

E

EDIT
system command, 69
EDT
system command, 73
EXECUTE
system command, 77

F

FIN
system command, 81

G

GLOBALS
system command, 83

H

HELP
system command, 87

I

INPL
system command, 89

K

KEY
system command, 91

L

LAST
system command, 95
LASTMSG
system command, 97
LIST
system command, 99
LIST XREF
system command, 143
LISTDBRM
system command, 135
LISTPROF
system command, 139
LISTSQL
system command, 145
LISTSQLB
system command, 149
LOGOFF
system command, 153
LOGON
system command, 155

M

MAIL
system command, 157
MAINMENU
system command, 159

N

NATQVS

system command, 161
NOCOPT
system command, 163
NOCSHOW
system command, 165
NOCSTAT
system command, 167

P

PROFILE
system command, 169
PROFILER
system command, 171

R

RDC
system command, 173
READ
system command, 175
RENAME
system command, 177
RENUMBER
system command, 179
RETURN
system command, 181
ROUTINES
system command, 183
RPCERR
system command, 185
RUN
system command, 187

S

SAVE
system command, 189
SCAN
system command, 191
SCRATCH
system command, 201
SETUP
system command, 203
SHOWDBS
system command, 207
SQLDIAG
system command, 211
SQLERR
system command, 215
STOW
system command, 217
STRUCT
system command, 219
SYSADA
system command, 227
SYSAPI
system command, 229
SYSBPM
system command, 231
SYSCP
system command, 233
SYSDB2
system command, 235
SYSDDM
system command, 237
SYSEDT
system command, 239
SYSERR
system command, 241
SYSEXT
system command, 243
SYSEXV
system command, 245
SYSFILE
system command, 247
SYSMAIN
system command, 249
SYSNCP
system command, 251
SYSOBJH
system command, 253
SYSPARM
system command, 255
SYSPROD
system command, 257
SYSPROF
system command, 261
SYSRPC
system command, 263
system commands, ix
SYSTP
system command, 265

T

TECH
system command, 267
TEST
system command, 269
TEST DBLOG
system command, 271

U

UNCATALOG
system command, 273
UNLOCK
system command, 275
UPDATE
system command, 281

W

WHICH
System command, 283

X

XREF
system command, 287

Z

ZIIP
system command, 289