Natural for Mainframes

System Commands

Version 4.2.6 for Mainframes (Update)

February 2010

Natural

This document applies to Natural Version 4.2.6 for Mainframes (Update).

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

Table of Contents

1 System Commands

This documentation describes the Natural system commands.

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

This documentation is organized under the following headings:

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 Function	Provides an overview of the Natural system commands grouped according to their functions.
System Commands in Alphabetical Order	Descriptions of the system commands in alphabetical order.

2 Issuing System Commands

Command Input

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

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

Element	Explanation
ABCDEF	Upper-case letters indicate that the term is either a Natural keyword or a Natural reserved word that must be entered exactly as specified.
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 entire bracketed expression.
,...	A term preceding a comma-ellipsis may optionally be repeated; if it is repeated, the repetitions must be separated by commas. A number after the comma-ellipsis indicates how many times the term may be repeated. If the term preceding the comma-ellipsis is an expression enclosed in square brackets or braces, the comma-ellipsis applies to entire bracketed expression.
:...	A term preceding a colon-ellipsis may optionally be repeated; if it is repeated, the repetitions must be separated by colons. A number after the colon-ellipsis indicates how many times the term may be repeated. If the term preceding the colon-ellipsis is an expression enclosed in square brackets or braces, the colon-ellipsis applies to entire bracketed expression.
Other symbols (except [ ] { } | ... ,... :...)	All other symbols except those defined in this table must be entered exactly as specified. **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 Function

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

Navigating within Natural

Command	Function
FIN	Terminates a Natural session.
LOGOFF	Causes the library ID to be set to SYSTEM and the Adabas password to be set to blanks. The contents of the source program work area are not affected by this command.
LOGON	Establishes a library ID for the user. In the specified library, all source or object programs saved during the session will be stored (unless you explicitly specify another library ID in a SAVE, CATALOG or STOW command).
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.

Environment Settings

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

Editing and Storing Programming Objects

Command	Function
CATALL	Catalogs *all* objects or selected objects in the current library.
CATALOG	Compiles the Natural programming object currently in the source work area of an editor, and if the syntax has been found to be correct, stores the resulting object module in the Natural system file.
CHECK	Checks that the source code of a programming object does not contain any syntax errors. The checking process varies according to the type of object being checked. Syntax checking is also performed as part of the system commands RUN, CATALL, CATALOG and STOW.
CLEAR	Clears the contents of the work area of the editor.
DELETE	Deletes a programming object from the Natural system file. You can delete it in source form, in object form, or both.
EDIT	Edits the source form of a programming object.
LIST	Lists one or more objects which are contained in the current library.
READ	Transfers an object in source form from the Natural system file to the source work area.
RENAME	Gives a Natural programming object another name.
RENUMBER	Renumbers the source code which is currently held in the source work area.
SAVE	Stores the *source form* of the programming object currently in the work area of the editor in the Natural system file.
SCAN	Searches for a string of characters within an object, with an option to replace the string with another string.
STOW	Compiles and stores a Natural programming object (in both source and object form) in the Natural system file.
STRUCT	Performs structural indentation of a source program, and helps detecting structural inconsistencies.

Executing Programs

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

Maintenance Utilities

Command	Function
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.
SYSNCP	Creates and maintains the command processors to be used in your Natural applications.
SYSRPC	Creates and maintains remote procedure calls, that is, provides the settings necessary to execute a subprogram located on a remote server.

Transfer of Programming Objects

Command	Function
SYSMAIN	Transfers objects within the Natural system from one library to another.
SYSOBJH	Processes Natural and non-Natural objects for distribution in Natural environments.

Monitoring and Debugging

Command	Function
BUS	Invokes the Buffer Usage Statistics function of the SYSTP utility. BUS provides statistical information on the usage of Natural buffers. **Note:** BUS replaces the system command SYSBUS which is no longer available.
DUMP	Provides information for Software AG technical support personnel in order to locate an error that caused an abnormal termination (abend) of the Natural system.
RPCERR	Displays the last Natural error number and message if related to Remote Procedure Call (RPC), and the last Broker reason code and associated message.

Command	Function
SYSADA (ADACALL)	Invokes the ADACALL utility which is contained in the library SYSADA. ADACALL enables you to issue Adabas direct calls (native commands) directly to an Adabas database.
SYSBPM	Invokes the SYSBPM utility, which is used to monitor the buffer pool and adjust it to meet your requirements.
SYSEDT	Invokes the Editor Buffer Pool Services utility SYSEDT, which is used to display runtime information of the editor buffer pool, modify its parameters, and delete logical work and recovery files.
SYSTP	Invokes the SYSTP utility, which allows you to monitor and control various 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 Natural DBLOG utility for logging database calls.

Commands Used with SQL Databases

Command	Function
LISTDBRM	Only available with Natural for DB2 and Natural for SQL/DS. Displays either existing DBRMs (Natural for DB2) or existing packages (Natural for SQL) of Natural programs or Natural programs referencing a given DBRM.
LISTSQL	Only available with Natural for DB2, Natural SQL Gateway, and Natural for SQL/DS. Generates a list of those Natural statements in the source code of a programming 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 SQL code before executing a Natural program which accesses an SQL table.
LISTSQLB	Only available with Natural for DB2. Invokes the function **Explain all SQL Statements**.
SQLDIAG	Only available with Natural for DB2. 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.
SQLERR	Only available with Natural for DB2, Natural SQL Gateway, and Natural for SQL/DS. Displays diagnostic information about the most recent SQL error.

Command	Function
SYSDB2	This command invokes Natural Tools for DB2, if Natural for DB is installed. For further information, see *Using Natural Tools for DB2* in the *Natural for DB2* part of the *Database Management System Interfaces* documentation.

Commands Used with the Natural Optimizer Compiler

The following system commands are only available if the Natural Optimizer Compiler is installed. For further information, refer to the *Natural Optimizer Compiler* documentation.

Command	Function
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.

Miscellaneous

Command	Function
AIV	Displays all application-independent variables (AIVs) that have been defined.
HELP	Invokes the Natural help system.
INPL	Invokes the INPL utility. It is *only* used for the loading of Software AG installation datasets into the system files.
LAST	Displays the system commands that were last executed, and allows you to execute them again.
LASTMSG	Displays additional information on the error situation which occurred last.
MAIL	Only available if Natural Security has been installed. 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.
ROUTINES	Shows you which objects in the current library use which external subroutines.
SYSEXT	Invokes the library SYSEXT, which contains various Natural user application interfaces.
SYSEXV	Invokes the SYSEXV application with examples of the new features of the current Natural versions.
SYSFILE	Invokes the function Natural Print/Work Files of the SYSFILE utility. This utility provides information on the work files and print files available.
UNLOCK	Enables you to view locked objects and unlock them if required.

Command	Function
UPDATE	Prevents database updating being carried out by a program.
XREF	Only available if Predict has been installed. Controls the usage of the Predict function “active cross-references”. This function automatically creates documentation in Predict about the objects which a program/data area references.

5 AIV

`AIV`

This command is used to display all 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

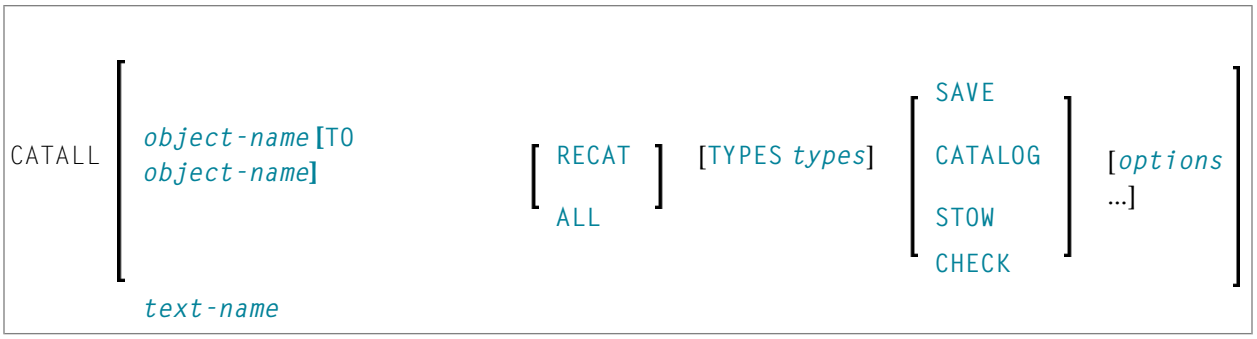
This command is used to invoke the function Buffer Usage Statistics of the SYSTP utility. It provides information on the buffers allocated for the current Natural session, their sizes, and the actual buffer space being used.

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.

Application Programming Interface: USR1019N. See *SYSEXT - Natural Application Programming Interfaces* in the *Utilities* documentation.

7 CATALL



This command is used to store all objects in the current library in source and/or object form.

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.

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 in source form 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**.

Recatalog Only Existing Modules, or Catalog All Sources

This option only applies to the functions Catalog and Stow:

- If you mark the first of the two fields, only those objects for which object modules already exist in the current library will be cataloged again; objects which only exist in source form will not be cataloged.
- If you mark the second of the two fields, *all* selected objects will be cataloged.

Note: This option does not apply to objects of type copycode and text.

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 modules 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 modules 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. Error Text Member: Enter the name of a Natural text member. An error list of a CATALL run will be written to this text member.

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*.
RECAT / ALL	Corresponds to the options **Recatalog Only Existing Modules**, or **Catalog All Sources** of the **Catalog Objects in Library** screen. RECAT is the default, see *Recatalog Only Existing Modules, or Catalog All Sources*.

<table>
<tr><td>TYPES types</td><td>Corresponds to the object types marked on the Catalog Objects in Library screen, see Select Object Types. Possible types are:

G Global data areas
A Parameter data areas
L Local data areas
C Copycodes
T Texts
S Subroutines
N Subprograms
H Helproutines
M Maps
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.</td></tr>
<tr><td>SAVE / CATALOG / STOW / CHECK</td><td>Corresponds to the actions of the same names on the Catalog Objects in Library screen, see Select Function. CATALOG is the default.</td></tr>
<tr><td>options</td><td>These options correspond to Select Options on the Catalog Objects in Library screen, see Select Options. Possible options are:
<table>
<tr><td>CC</td><td>Condition Code will be returned.</td></tr>
<tr><td>NOREN</td><td>No automatic renumbering of source-code lines.</td></tr>
<tr><td>KEEP</td><td>Result list will be kept.</td></tr>
<tr><td>NOSCROLL</td><td>Online: scrolling display of processing status information will be suppressed. Batch: output of only those modules, which caused an error.</td></tr>
<tr><td>NOREPORT</td><td>Error report will be suppressed.</td></tr>
<tr><td>FULL</td><td>Error report will be extended.</td></tr>
<tr><td colspan="2">EL <<i>text-member</i>> [R]</td></tr>
<tr><td>EL <<i>text-member</i>></td><td>Output error list to a Natural text member.</td></tr>
<tr><td>R</td><td>If a member already exits, the EL parameter is disabled, unless R (replace) is specified behind <<i>text-member</i>>.</td></tr>
<tr><td colspan="2">Note: If you specify NOREPORT and NOSCROLL, KEEP will automatically apply, too.</td></tr>
</table></td></tr>
<tr><td>text-name</td><td>Corresponds to specifying a text name in the Catalog Objects from field of the Catalog Objects in Library screen, see Catalog Objects from/to.</td></tr>
</table>

Examples:

To stow only objects for which object modules 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 | UNCATALOG.

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

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 CMS

This command applies only to VM/CMS environments and allows the execution of CMS commands from within Natural.

```
CMS CMS-command
```

The command specified in place of *CMS-command* is passed to the CMS command processor which provides full CMS command resolution just as in normal CMS interactive command mode.

For further information on CMS, see *Natural under VM/CMS* (in the *Operations* documentation).

12 COMPOPT

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

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

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

The default settings of the individual options are set with the corresponding 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.

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 NATPARM.
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 programming 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 programming object, but do not update settings set with the command `COMPOPT`.

General Compilation Options

- KCHECK - Keyword Checking
- PCHECK - Parameter Checking for Object Calling Statements
- DBSHORT - Interpretation of Database Short Field Names
- PSIGNF - Internal Representation of Positive Sign of Packed Numbers
- TSENABL - Applicability of TS Profile Parameter
- GFID - Generation of Global Format IDs
- LOWSRCE - Allow Lower-Case Source
- TQMARK - Translate Quotation Mark
- THSEP - Dynamic Thousands Separator
- CPAGE - Code Page Support for Alphanumeric Constants
- DB2ARRY - Support DB2 Arrays in SQL SELECT and INSERT Statements
- CHKRULE - Validate INCDIR Statements in Maps

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

KCHECK - Keyword Checking

ON	Field declarations in a programming 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 programming 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.

PCHECK - Parameter Checking 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, 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. **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 programming 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, external subroutines, subprograms, helproutines, 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 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 view field is used in the programming code; ■ when a DEFINE DATA LOCAL statement was previously used to defines variables; ■ 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
```

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'.

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.

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=abbbbbbc` for file number less than or equal to 255 and Adabas Version lower than 6.2 (see `NTDB` macro).

where	equals
a	x'F8' or x'F7' or x'F6'
bbbbbb	bytes 1-6 of STOD value
c	physical file number

  - `GFID=axbbbbbc` for file number greater than 255 and Adabas Version lower than 6.2.

where	equals
a	x'F8' or x'F7' or x'F6'
x	physical file number - high order byte
bbbbb	Bytes 2-6 of STOD value
c	physical file number - low order byte

  - `GFID=abbbbbb` for Adabas Version 6.2 or higher.

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).

LOWSRCE - Allow Lower-Case Source

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

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
```

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.

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.

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.

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 programming 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.

Compilation Options for Ensuring Version Compatibility

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

FINDMUN - Detect Inconsistent Comparison Logic in FIND Statements

With Natural Version 2.3, the comparison logic for multiple-setting fields in the WITH clause of the FIND statement has been changed. This means that when Version 2.2 programs containing certain forms of FIND statements are compiled under Version 3.1, they will return different results. This option can be used to search for FIND statements whose WITH clauses use multiple-setting fields in a way that is no longer consistent with the enhanced Version 3.1 comparison logic.

ON	Error NAT0998 will be returned for every FIND statement of such form detected at compilation.
OFF	No search for such FIND statements will be performed. This is the default value.

The comparison logic for multiple-value fields in the WITH clause of the FIND statement has been changed with Natural Version 2.3 so as to be in line with the comparison logic in other statements (e.g. IF).

Four different forms of the FIND statement can be distinguished (the field MU in the following examples is assumed to be a multiple-value field):

1.
```
FIND XYZ-VIEW WITH MU = 'A'
```

   With Version 2.2 and above, this statement returns records in which at least one occurrence of MU has the value "A".

2.
```
FIND XYZ-VIEW WITH MU NOT EQUAL 'A'
```

   With Version 2.2, this statement returns records in which no occurrence of MU has the value "A" (same as 4.). With Version 2.3 and above, this statement returns records in which at least one occurrence of MU does not have the value "A".

3. `FIND XYZ-VIEW WITH NOT MU NOT EQUAL 'A'`

   With Version 2.2, this statement returns records in which *at least one occurrence* of `MU` has the value "A" (same as 1.). With Version 2.3 and above, this statement returns records in which *every occurrence* of `MU` has the value "A".

4. `FIND XYZ-VIEW WITH NOT MU = 'A'`

   With Version 2.2 and above, this statement returns records in which *no occurrence* of `MU` has the value "A". This means that if you newly compile under Version 2.3 existing Version 2.2 programs containing `FIND` statements of the forms 2. and 3., they will return different results.

If you specify `FINDMUN=ON`, error NAT0998 will be returned for every `FIND` statement of form 2. or 3. detected at compilation.

Should you in these cases wish to continue to get the same results as with Version 2.2, you have to change the statements as follows:

In Form 2:

```
FIND XYZ-VIEW WITH MU NOT EQUAL 'A'
```

into

```
FIND XYZ-VIEW WITH NOT MU = 'A'
```

In Form 3:

```
FIND XYZ-VIEW WITH NOT MU NOT EQUAL 'A'
```

into

```
FIND XYZ-VIEW WITH MU = 'A'
```

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 to `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.

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.

V41COMP - Disable New Version 4.2 Syntax

Important: This compiler option will be available only with Natural Version 4.2 to allow a smooth transition. It will be removed again with a subsequent release of Natural after Version 4.2.

A number of functions and programming features introduced with Natural Version 4.2 would give rise to problems when a program developed and compiled with Version 4.2 is to be recompiled for putting into operation in a Version 4.1 environment. The relevant functions or features are listed **below**.

The `V41COMP` option has been provided to detect such incompatibilities and trigger an error message that supplies a reason code for why the recompilation failed. The following values are possible:

ON	When a program is compiled under Version 4.2, every attempt to use a syntax construction that is supported by Version 4.2, but not by Version 4.1, is rejected and a NAT0647 syntax error and a corresponding reason code (see **below**) will be output.
OFF	A test for Version 4.1 compatibility is not performed. This is the default value.

Compilation Relevant Differences between Version 4.2 and 4.1

The following table gives an overview of the compilation relevant differences between Version 4.2 and 4.1 and indicates the reason code that will be supplied when incompatible syntax is detected:

Function or Feature	Version 4.2	Version 4.1	Reason Code
New format U (Unicode)	possible	unknown	001
Array with variable number of occurences	possible	unknown	002

Function or Feature	Version 4.2	Version 4.1	Reason Code

X-array, for example:
 	
`DEFINE DATA LOCAL`
`1 #ARR (A10/1:*)`			

Possible length of alpha and literals (constants)	1 byte - 1 GB	1 byte - 253 bytes (NAT0264)	003
New compiler parameters: `THSEP` Thousands separator character in edit mask `CPAGE` Make alphunmeric constants sensitive for code page translation	possible	unknown	004

New statements:
 	
`MOVE NORMALIZED`
`MOVE ENCODED`
`PARSE`
`REQUEST DOCUMENT`
`EXPAND / REDUCE / RESIZE ARRAY`
	possible	unknown	005

Statement `SET GLOBALS`: ■ session parameter `CPCVERR=ON/OFF` ■ allowed when in structured mode (`SM=ON`)	possible	unknown	006

New system variables:
 	
`*PARSE-COL`
`*PARSE-LEVEL`
`*PARSE-NAMESPACE-URI`
`*PARSE-ROW`
`*PARSE-TYPE`
`*CODEPAGE`
`*LOCALE`
`*TYPE`
`*CURRENT-UNIT`
`*UBOUND`
`*LBOUND`
	possible	unknown	007

Not used	-	-	008
Length and type of source parameters supplied with `INCLUDE`	any length and format U (Unicode) allowed	only alpha with a length of max. 80 bytes	009

Function or Feature	Version 4.2	Version 4.1	Reason Code

Example:
 	
`INCLUDE COPY01 'WRITE *LINE'`
`'WRITE *PROGRAM'`			

Definition of an Adabas LA-field in a data view ■ with a size greater than 253 bytes or ■ of type `DYNAMIC`	possible	unknown	010

13 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, `NATICU` version, Unicode version, etc., and to display the code pages defined in the `NATCONFG` module. For full details, see the corresponding Natural Help text.

For further information, see the *Unicode and Code Page Support* documentation.

14 DELETE

```
DELETE [ [TYPE object-type ...] [ { SOURCE } ] object-name ]...
                                { OBJECT }
                                { BOTH   }
```

This command is used to delete Natural programming 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.

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 an object's source code, its object module, or both are to be deleted: `SOURCE` — source code `OBJECT` — object module `BOTH` — both, source code and object module. 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 its source code, its object module, or both - by marking the object with the appropriate letter (S , O or B).

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.

Examples

With this command, you delete three programming objects named TOM, DICK und HARRY:

```
DELETE TOM DICK HARRY
```

With this command, you delete the source and object module of the programming object JOHN, the sources of the programming objects PAUL and GEORGE, and the object module of the programming object RINGO:

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

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

```
DELETE
```

With this command, you get a selection list of the sources 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 in source and/or object form and whose names begin with D:

```
DELETE TYPE GLA D*
```

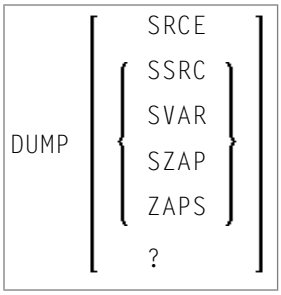
With this command, you get a selection list of all programming objects in the current library which are stored in object form and whose names begin with "YYZ":

```
DELETE OBJECT YYZ*
```

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

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

15 DUMP



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

DUMP	Displays abend information (core contents).
DUMP SRCE	Displays the inventory of source changes applied per product.
DUMP SSRC	Displays the inventory of special source changes applied per product.
DUMP SVAR	Displays TP monitor and operating system dependent system variables and additional information.
DUMP SZAP	Displays a list of all special Zaps applied.
DUMP ZAPS	Displays a list of all Zaps applied.
DUMP ?	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, Software AG technical support personnel will tell you when and how to use these options.

16 EDIT

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

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

Related command: READ.

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

Syntax 1

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

object-type

The following object types can be edited:

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

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 with the program editor.
- All other types of objects - program, subprogram, subroutine, helproutine, copycode, text, description - are edited with the program editor.

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

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

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 member 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.

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.

Syntax 3

```
EDIT FUNCTION subroutine-name
```

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

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
```

17 EDT

It is recommended that the EDIT command be used 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.

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 program 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.
.E	Exit from line editor.

Command	Function
.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

The following PF keys can be used during line editing:

Key	Command	Function
PF1	.-18	Scroll 18 lines backwards.
PF2	.T	Scroll to top.
PF3	.B	Scroll to bottom.
PF4	.+5	Position 5 lines forwards.
PF5	.+10	Position 10 lines forwards.
PF6	.+18	Position 18 lines forwards.
PF7	.R	Renumber.
PF8	.I	Insert line.
PF9	.E	Exit from line editor.
PF10	.E,RUN	Exit from line editor and run program.
PF11	.E,SAVE,RUN	Exit from line editor, save and run program.
PF12	.E,CAT,SAVE,EX	Exit from line editor, catalog, save and execute program.

18 EXECUTE

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

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

Syntax Explanation

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
```

19 FIN

FIN

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

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

20 GLOBALS

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

This command is used to set Natural session parameters.

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=.`

List of Parameters

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

Parameters	Function
CC	Error Processing in Batch Mode
CF	Character for Terminal Commands
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

Parameters	Function
IM	Input Mode
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.

21 HELP

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

HELP	Displays the help menu.
HELP *statement*	Displays help information for the specified *statement*.
HELP *command*	Diplays help information for the specified *command*.
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.

22 INPL

`INPL [R]`

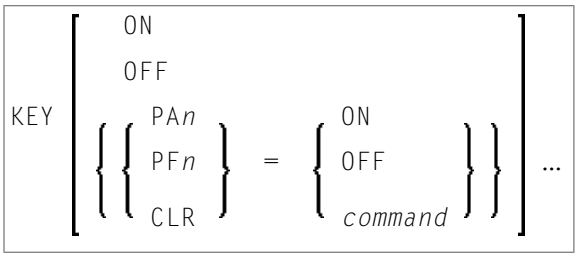
This command is used to invoke the Natural `INPL` utility. This utility is *only* used for the loading of Software AG installation datasets into the system files as described in the `INPL` online help and in the platform-specific installation documentation.

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

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` to its original status. Then the user DBA, the library `SYSSEC`, and the link between the two will be redefined as after the initial installation, while all other links to `SYSSEC` will be cancelled. See also *Inaccessible Security Profiles* in the section *Countersignatures* of the *Natural Security* documentation.

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

23 KEY



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.

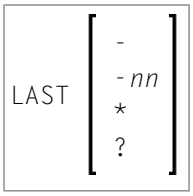
Activating/Deactivating Individual Keys - KEY key=ON/OFF

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 command KEY CLR=ON/OFF is not possible (see also note **above**).

24 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.

25 LASTMSG

LASTMSG

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

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

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

To display information on the corresponding error

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

  The following is displayed:

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

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

Natural Remote Procedure Call (RPC):

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

26 LIST

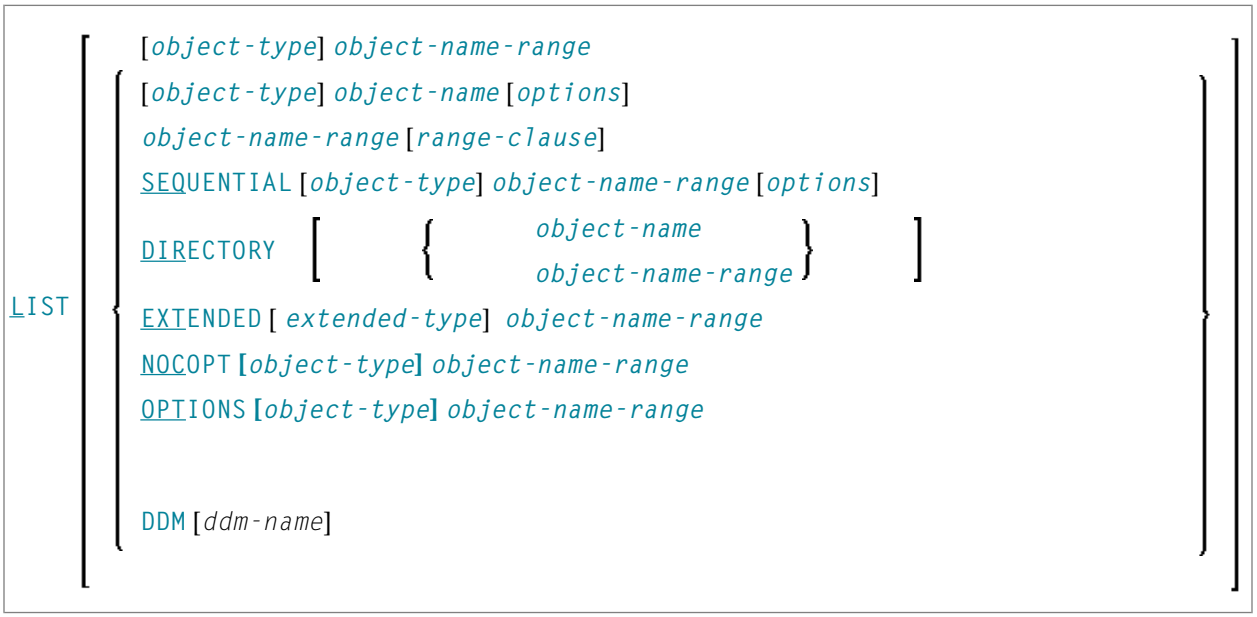
This system command is used to display the source code of a single object or to list one or more objects which are contained in the current library. The options of the LIST command are explained below.

This chapter covers the following topics:

See also separate documents describing LIST XREF, LISTDBRM, LIST COUNT and LISTSQL.

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

Syntax Overview

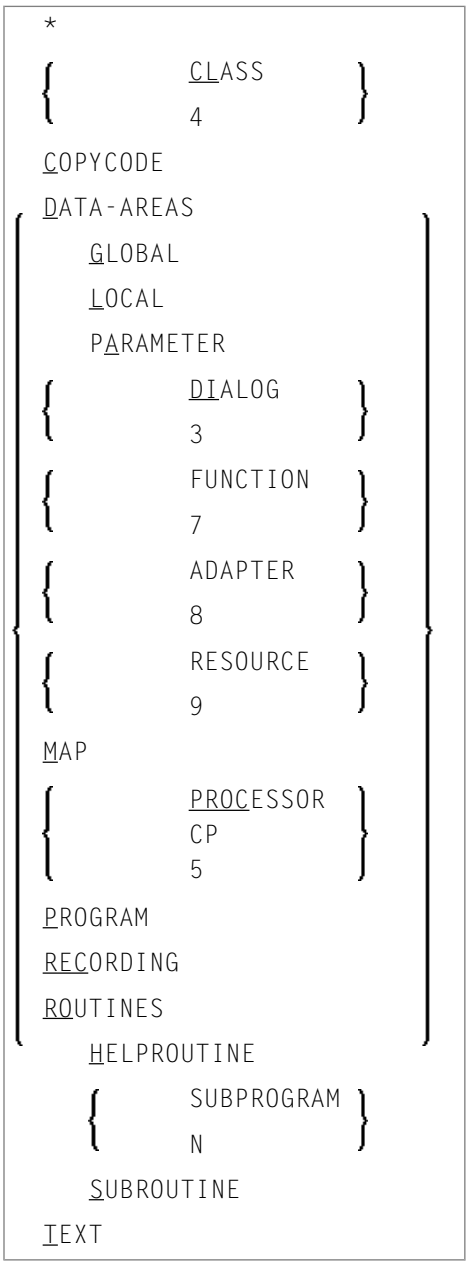


Notes:

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:

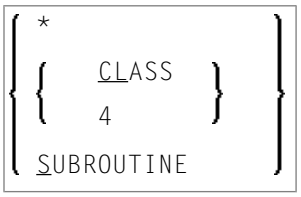
- To have all objects in the current library listed, you specify an asterisk (*) for the *object-name-range*, but no *object-type*.
- To have all objects of a certain type listed, you specify a certain *object-type* and an asterisk (*) for the *object-name-range*.
- If you wish a certain range of objects 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*** .

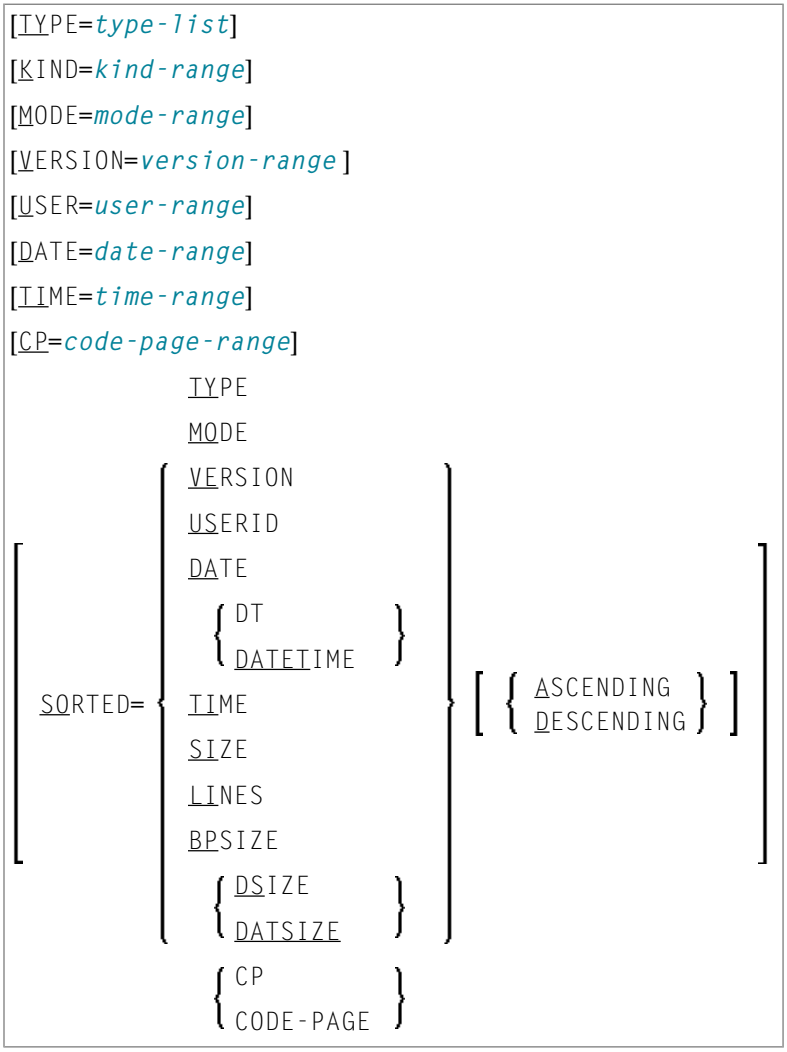
extended-type

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



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. See also the definition of the term Version in the Glossray. Valid version format: *V.R.SM* where *V* is the 1-digit version, *R* the 1-digit release, and *SM* the 2-digit system maintenance level. You can also specify a range of versions: see range-notation.	
user-range	The ID of the user who saved or cataloged a Natural programming object. Specify a single user ID or a range of user IDs: see *range-notation* below.	
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 catalogued within the last 2 days.
	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	Specify a single code page or a range of code pages: see *range-notation* below.

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 notation > or < respectively.
- The notations < and > cannot be combined with each other or with asterisk or wildcard notation.

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 Sources Sequentially

LIST SEQUENTIAL *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.
LIST SEQUENTIAL *object-type object-name-range* **[***options***]**	

Displaying a List of Objects

LIST *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***).
LIST *object-type object-name-range*	

Displaying a Presorted List of Preselected Objects

LIST *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***). With the *range-clause*, you specify additional selection and sorting criteria. See also **example** below.
LIST *object-name-range range-clause*	

Displaying Long Names of Cataloged Subroutines and Classes

LIST EXTENDED *object-name-range*	Displays a list of the long names of cataloged subroutines and classes. 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 Directory Information

LIST DIRECTORY	Displays the directory information on the object currently in the work area: ■ **Source code:** "Saved-on" date and time, library name, user ID, programming mode (reporting or structured), TP system, terminal ID, operating system, transaction, Natural version, code page information (if available), source size ■ **Object code:** "Cataloged-on" date and time, library name, user ID, programming mode, TP monitor system, terminal I/O, transaction, Natural version, code page information (if available), operating system/version, GDA used, size of global data, size in DATSIZE, size in buffer pool, size of OPT-code (size of machine code generated by Natural Optimizer Compiler), initial OPT-string (OPT profile parameter value effective at STOW time), compiler options
LIST DIRECTORY *object-name*	Displays the directory information (as described for LIST DIRECTORY) on the specified object.
LIST DIRECTORY *object-name-range*	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 *object-name* **WITH DIRECTORY**	This command first displays the directory information (as described for `LIST DIRECTORY`) on the specified object and then lists the source code of the object.

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).

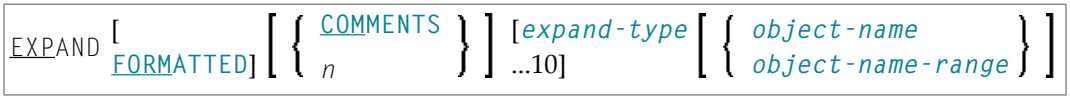
Options

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

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

DIRECTORY	This option first displays the directory information (as described below, see ***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*.)
CONVERTED	By default, the source is listed in the code page as stored on the system file. To list the source in the default code page (see system variable `*CODEPAGE`), specify this option.

expand-option



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).
EXPAND COMMENTS **EXPAND** *n*	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.

<table>
<tr><td></td><td colspan="3">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></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></td><td>N</td><td>Subprograms</td></tr>
<tr><td></td><td>S</td><td>External subroutines</td></tr>
<tr><td></td><td>H</td><td>Helproutines</td></tr>
<tr><td></td><td>G</td><td>Global data areas</td></tr>
<tr><td></td><td>L</td><td>Local data areas</td></tr>
<tr><td></td><td>A</td><td>Parameter data areas</td></tr>
<tr><td></td><td>M</td><td>Maps</td></tr>
<tr><td></td><td>C</td><td>Copycodes</td></tr>
<tr><td></td><td>E</td><td>Error messages</td></tr>
<tr><td></td><td>4</td><td>Classes</td></tr>
<tr><td></td><td>*</td><td>All object types</td></tr>
<tr><td>object-name</td><td colspan="3" rowspan="2">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>
<tr><td>object-name-range</td></tr>
</table>

formatted-option

```
FORMATTED ['c']  ['c']  [SETTINGS] [ { FIELDS    } ] [ { RULES       } ]
                                   [ { EXTFIELDS } ] [ { INLINERULES } ]
                                                     [ { FREERULES   } ]
                                                     [ { AUTORULES   } ]
```

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; that is, 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	
CODE-PAGE or CP	ON	Display the code page information for each object. This is the default value.
	OFF	Do not display the code page information.
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; same as LIST EXTENDED *.	
ALL *fx*	Enter the function code *fx* (where *fx* is a valid function code for a listed object) for all displayed objects.	
SORT	Invokes 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 *dataarea* 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.

Code	Function	
LISTPROFILE	Display the current value of the parameters of the LIST profile (see *Defining an Individual List Profile* below).	
NOCOPT	Displays 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	Displays 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*.	
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) 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 mainframe to personal computer (only available if Natural Connection is installed).
ED	Edit the object's source code (equivalent to the system command EDIT).
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.
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).

Code	Function
.	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.

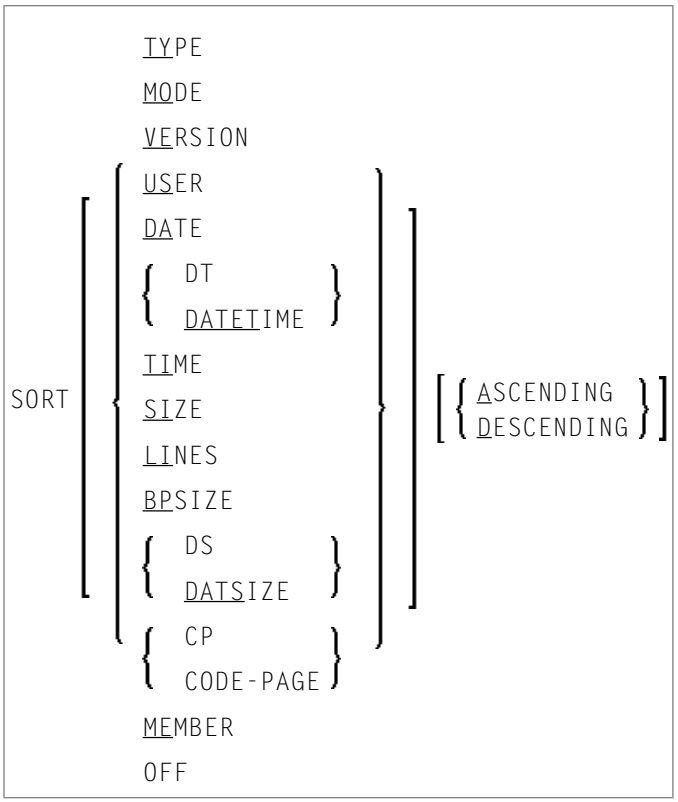
To invoke the sort function

- 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:

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
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
Member names of subroutines or classes (available in extended selection list only)	MEMBER

Once the sort has been started, all changes in the Criteria for the Selection List create a sorted list.

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 contains 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`* stands for the relevant version and release numbers).
`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 of Source

The following topics are covered below:

- Cursor-Sensitive Object Selection

Subcommands for Listed Source

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

Subcommand	Function
+	Scrolls one page forward.
-	Scrolls one page backward.
++	Scrolls to the end (bottom) of the source.
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*.
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*. If you enter only the command FIND itself, a window will be displayed in which you can enter the *value* to be sought and specify whether the search is to be absolute or not. If you specify FIND without ABSOLUTE after the command, the *value* will only be found if it is an isolated word. This is the default. If you specify ABSOLUTE after the command, the *value* will also be found if it is part of a larger string of characters.
FIND *value*	
FIND ABSOLUTE *value*	
FORMAT	Applies to data areas and maps only: displays ***formatted*** data area or map, and other items related to the map.
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.

Subcommand	Function
PRINT	Prints the listed source.
REF REF *value* REF ABSOLUTE *value*	Displays the line numbers of the source-code lines which contain the specified *value* in a table. If you enter only the command REF itself, a window will be displayed in which you can enter the *value* to be sought for and specify whether the search is to be absolute or not. If you specify REF without ABSOLUTE after the command, the *value* will only be found if it is an isolated word. This is the default. If you specify ABSOLUTE after the command, the *value* will also be found if it is part of a larger string of characters.
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).
SCAN SCAN *value* SCAN ABSOLUTE *value*	Displays all lines intensified which contain the specified *value*. The source will be scrolled to the first line that contains the *value*. If you enter only the command SCAN itself, a window will be displayed in which you can enter the *value* to be sought for and specify whether the search is to be absolute or not. If you specify SCAN without ABSOLUTE after the command, the *value* will only be found if it is an isolated word. This is the default. If you specify ABSOLUTE after the command, the *value* will also be found if it is part of a larger string of characters.
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* ZOOM [*expand-type*...10] *object-name-range*	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.

Subcommand	Function
.	Exit.

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 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

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, Natural provides the text object LISTPROF in the library SYSLIB.

In LISTPROF, you can enter general or user-specific profiles with corresponding defaults, such as COUNT-SOURCE-LINES. These defaults are used when you start the LIST command.

To activate the values defined in LISTPROF

1 Copy the text object LISTPR-S from library SYSLIB to any library.

2 Add the changes.

3 Save the text object LISTPR-S under the name LISTPROF.

4 Copy the text object LISTPROF to library SYSLIB.

5 Invoke the LIST command.

For a detailed description, see text object LISTPR-S in library SYSLIB.

27 LISTDBRM

`LISTDBRM`

Notes:

1. This command is only available with Natural for DB2 and Natural for SQL/DS.
2. In case of Natural for DB2, `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`.
3. In case of Natural for SQL/DS, `LISTDBRM` has to be issued from library `SYSSQL`, which means you have to log on to `SYSSQL` first and then enter the command `LISTDBRM`.

The `LISTDBRM` command is used to display either existing DBRMs (Natural for DB2) or existing packages (Natural for SQL) of Natural programs or Natural programs referencing a given DBRM.

When the command is used with Natural for DB2 or Natural for SQL/DS, 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 (Natural for DB2) or SQL/DS (Natural for SQL/DS), 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.
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
```

28 LIST COUNT

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

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

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

29 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.

30 LISTSQL

This command is only available with Natural for DB2, Natural SQL Gateway, and Natural for SQL/DS. There are minor differences depending on whether the command is used with Natural for DB2, Natural SQL Gateway, or Natural for SQL/DS. These differences are marked accordingly in the following description.

```
LISTSQL [ { object-name } [ALL] ]
        [ { <sa>        }       ]
```

This command generates a list of those Natural statements in the source code of a programming 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 SQL code 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 4.10
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
                                                                  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. A static DBRM is only available with Natural for DB2 and Natural for SQL/DS.

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 or SQL/DS errors, respectively.
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 and Natural for SQL/DS.
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 Parameters Screen:

```
14:55:24               ***** NATURAL TOOLS FOR SQL *****           2009-12-04
Member DEM2SEL                      LISTSQL                    Library SYSDB243

        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 or SQL/DS consistency token, and some internal static parameters.

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 *Special Requirements for Natural Tools for DB2* in the *Natural for DB2* 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 inform-

ation 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.

Using the EXPLAIN Command with Natural SQL Gateway

This command is not applicable.

Using the EXPLAIN Command with Natural for SQL/DS

LISTSQL enables you to use the SQL/DS command EXPLAIN, which provides information on the SQL/DS optimizer's choice of strategy for executing SQL statements.

Natural executes the EXPLAIN command for the SQL statement that is displayed on the LISTSQL screen.

The information determined by the SQL/DS optimizer is written into your PLAN_TABLE. Natural then reads the table and displays the contents.

Sample **EXPLAIN Result** Screen:

```
 10:22:07               ***** NATURAL TOOLS FOR SQL *****            2009-12-03
 Queryno 1                      EXPLAIN Result                      Row  1 / 1

                      Estimated cost :     3.3  timerons

            Qblockno                   Table
                  Planno Method Tabno creator     Tablename
              ---  ---   ------  ---  -------- ------------------
              1    1             1    NAT      DEMO




           Access  Access
            type   creator      Accessname      sort_new sort_comp
            ----   --------  ------------------ -------- ---------
             R                                    N          N




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

31 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 *Special Requirements for Natural Tools for DB2* in the *Natural for DB2* 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 dataset referred to by DD name `CMPRINT` (logical printer 0).

32 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.

33 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.

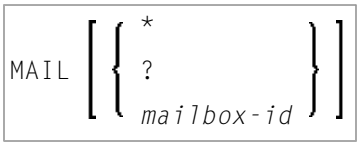
LOGON causes all Natural global data areas and application independent variables (AIVs), all assignments made using the SET KEY statement and retained ISN lists to be released. Data definition modules (DDMs) contained in the DDM buffer area are also released.

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.

34 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.

35 MAINMENU

This command is used to activate/deactivate Natural main menu mode.

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

MAINMENU	Switches main menu mode on. This is the default.
MAINMENU ON	
MAINMENU OFF	Switches main menu mode off.
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.

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 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.

41 RENAME

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

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

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

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

RENAME	If you issue the command without parameters, a **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:

4	Class
5	Processor
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

42 RENUMBER

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

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

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

43 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.

44 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 name of the subroutine, class or function it contains is displayed.

45 RPCERR

RPCERR

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

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

46 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).

47 SAVE

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

Related commands: STOW | CATALOG.

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

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.

48 SCAN

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.

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

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

This chapter covers the following topics:

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; ■ the number of objects in which the scan value was found; ■ the number of source-code lines in which the scan value was found.
	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 **subcommands** 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.
	S	**Object Lines with Scan Value**

Field	Input setting	
		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 **subcommands** 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 and functions 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. See also *Object Naming Conventions* in the *Using Natural* documentation.	
Object type(s)	You can restrict the search to specific object types. 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. **Note:** In data areas, a scan is always absolute, regardless of the value of this parameter.

Field	Input setting	
Selection list	Y	Displays 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) for scan processing.
	N	By default, no selection list is displayed.
Trace	Y	Activates the trace facility.
	N	By default, the trace facility is deactivated.

SCAN Subcommands

The following subcommands can be entered in the command line(s) of the result screens generated by the scan operation:

Command	Function
blank	Continue with normal scan processing.
`Q`	Terminate scan processing.
`.`	
`E`	Edit the object using full-screen editor.
`EDT`	Edit the object using line editor.
`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. A "D" will appear next to the line to indicate that it has been deleted.
`.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 dot (`.`) is specified before the next object is scanned.
- Lines containing `PASSW`, `PASSWORD=`, `CIPHER=`, or `CIPH=` will be ignored.

SCAN Keywords

SCAN functions may be invoked direct, in either batch or online mode, by specifying the following keywords:

Keyword	Explanation
FUNC	Function code
SVAL	Scan value
LIB	Library
RVAL	Replace value
OBJ	Object name
TYPE	Object type
ABSOL	Absolute scan

Caution: In a direct command, values containing embedded blanks must be avoided to prevent undesired scan/replace results. In online mode, a scan for a value that contains embedded blanks is only possible from the **SCAN menu**.

Examples of SCAN Command with Keywords:

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

```
SCAN FUNC=S,SVAL=value,RVAL=value,OBJ=PGM1
```

SCAN in Batch Mode

The SCAN command will process only one function per invocation to minimize the repercussions of invalid data being specified. Either keywords (as described **above**) or positional parameters may be used.

Positional parameters are specified as follows:

```
SCAN func, scan-value, replace-value, library, object-name, object-type, absolute
```

Possible values of the positional parameters are as decribed under ***Menu Options***.

Important: To scan for a value that contains lower-case characters or embedded blanks, do not specify the *scan-value* in the same line of the batch job as the SCAN command, but in a separate data line, and enter the data according to the online map, see ***Menu Options***.

Example of using a scan/replace value with embedded blanks:

```
SCAN S,MOVE LEFT,MOVE RIGHT,SYSTEM,PGM0*,N,*,N,N
```

SCAN under Natural Security

For you to be able to use the SCAN in a Natural Security environment, the system commands LIST, EDT, EDIT, and READ must be allowed in the current library's security profile. If the **Replace value** option is to be used or if the source is to be modifiable, the system command SAVE must also be allowed.

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

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

49 SCRATCH

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

50 SETUP

```
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.

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.

51 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 diagonstics information see the IBM DB2 documentation of the GET DIAGOSTICS statement.

Fields, which are prefixed with a 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 abreviation of the field (which are the capital letters of the description) prefoxed by the '+' sign in the command line. For example,. +SN shows a window with the full value of the 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** Screens:

```
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
```

52 SQLERR

SQLERR

Notes:

1. This command is only available with Natural for DB2, Natural SQL Gateway, and Natural for SQL/DS.
2. There are minor differences depending on whether the command is used with Natural for DB2, Natural SQL Gateway, or Natural for SQL/DS. These differences are marked accordingly in the following description.

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);
- for Natural for DB2: the variables SQLSTATE and SQLCODE returned by DB2
- for Natural SQL Gateway: the SQL code returned by the ConnecX SQL engine or the SQL database system;
- for Natural for SQL/DS: the variables SQLSTATE and SQLCODE returned by SQL/DS;
- 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   SQL code : -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
```

Sample SQLERR Diagnostic Information Screen (Natural SQL Gateway)

```
                 *** SQLERR Diagnostic Information ***
---------------------- Natural SQL Interface Codes -------------------------
Return Code: 3700    Reason Code: 0                 ZZN01   SQL code: -4017
--------------------------------- SQLCA -------------------------------------
SQLERRD (Additional Information)
        Number of Rows Processed                           :           0

SQLWARN (Warning Flags)
        Data truncated                                     :

        No. of columns greater than No. of host variables  :
CNX Error Message :
4017(E): SERVER ERROR: ODBC:(HY000) NATIVE:(0) : Ambiguous table reference: (D
EMO) ? PERS_ID , NAME , ADDRESS , DATEOFBIRTH , SALARY FROM << Syntax Error >>
  NSB.DEMO ?.
```

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

Sample SQLERR Diagnostic Information Screen (Natural for SQL/DS)

```
                  *** SQLERR Diagnostic Information ***
---------------------- NATURAL SQL Interface Codes -------------------------
Return Code: 3700              Reason Code: 0               SQL code : -204
------------------------------- SQLCA ----------------------------------------
SQLERRP (Adabas SQL Subroutine where error occurred)       :  ARIXOCA
SQLERRD (Adabas SQL Internal State)
        RDS Return Code                                    :            100
        DBSS Return Code                                   :              0
        Number of Rows Processed                           :              0
        Estimated Cost                                     :              1.0
        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 causes a performance degradation     :
        Adjustment to DATE/TIMESTAMP Variable made         :
SQL/DS Error Message :
SAG.SYSTABLES  not found in system catalog
```

53 STOW

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

Related commands: SAVE | CATALOG.

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

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.

54 STRUCT

STRUCT

This command serves two purposes:

- You can use it to perform structural indentation of the source code of the programming 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.

55 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 mainframe Natural.

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.

56 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.

57 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.

58 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.

59 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.

60 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.

61 SYSEDT

SYSEDT

This command is used to invoke the SYSEDT utility for Editor Buffer Pool Administration. 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.

62 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.

63 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.

64 SYSEXV

SYSEXV

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

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

65 SYSFILE

```
SYSFILE [ { WORKFILE } ]
        [ { PRINTER  } ]
```

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.

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.

66 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.

67 SYSNCP

SYSNCP

This command is used to invoke the SYSNCP utility.

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

68 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.

69 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.

70 SYSPROD

SYSPROD

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

Application Programming Interfaces: USR0050N, USR2031N. See *SYSEXT - Natural Application Programming Interfaces* in the *Utilities* documentation.

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

- the product name,
- the product version and release,
- the system maintenance (SM) level,
- the installation date and time.

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

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

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

SYSPROD Command Interface (Batch)

For batch processing or for an unformatted online output, you may call `SYSPROD` with additional command line parameters:

```
SYSPROD { ALL              } [EX] [SC] [HI]
        { <product-code>   }
```

71 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.

72 SYSRPC

SYSRPC

This command is used to invoke the SYSRPC utility.

The SYSRPC utility provides functions for maintaining remote procedure calls.

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

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

73 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.

74 TECH

TECH

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

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

Notes:

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

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

75 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 mainframes 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 UNIX or OpenVMS) local environment. If there is an active connection to a development server on a mainframe, the mainframe utility TEST is invoked when this command is issued under Natural for Windows.

76 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 a mainframe, the Natural mainframe utility DBLOG is invoked when this command is issued under Natural for Windows.

77 UNCATALOG

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

78 UNLOCK

This command is used for local unlocking of Natural source objects in a Natural mainframe environment.

It is used to view source objects that are locked and to unlock them if need be. 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 map 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]]
```

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)

Note: Locking can also be enabled locally on a mainframe server based on Natural for Mainframes Version 4.2 or above. 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 with an error message box will appear.

In the following cases, the locked objects found are listed in a 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.

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.

79 UPDATE

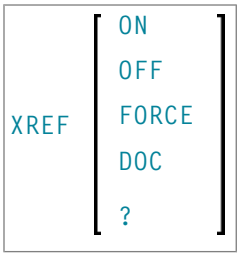
```
UPDATE { ON  }
       { OFF }
```

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

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

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

80 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.

Index

S