

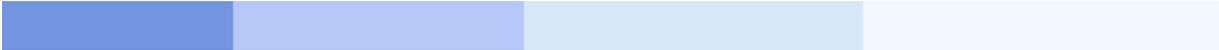


NATURAL

Natural

Utilities

Version 3.1.6 for Mainframes



This document applies to Natural Version 3.1.6 for Mainframes and to all subsequent releases. Specifications contained herein are subject to change and these changes will be reported in subsequent release notes or new editions.

© June 2002, Software AG
All rights reserved

Software AG and/or all Software AG products are either trademarks or registered trademarks of Software AG. Other products and company names mentioned herein may be the trademarks of their respective owners.

Table of Contents

Natural Utilities for Mainframes - Overview	1
Natural Utilities for Mainframes - Overview	1
NATRJE - Natural Remote Job Entry	2
NATRJE - Natural Remote Job Entry	2
NATRJE General Information	2
Calling NATRJE from a Natural Program	2
Invoking NATRJE	3
Example Programs	3
NATRJE Return Codes	6
Return Codes Common to all Environments	6
Additional Return Codes for VSE/ESA	6
Additional Return Codes for BS2000/OSD	7
NATRJE User Exit	7
NATRJE Features Applicable to UTM/TIAM	7
Additional Values for the parm3 Parameter	8
Name of BS2000/OSD Dataset	8
SYSDDM Utility	9
SYSDDM Utility	9
DDMs	9
SYSDDM and Predict	9
Invoking SYSDDM	10
Overview of Functions	11
Generate DDM from Adabas FDT	13
Catalog DDM	13
Edit DDM	13
DDM Editor	14
Field Attributes	15
DDM Editor Commands	17
Extended Field Editing	19
Delete DDM	20
List DDMs	20
List DDMs with Additional Information	20
Copy DDM to Another FDIC File	21
Show Defined DBIDs and Used FNRs	21
Database IDs Defined in Natural	21
File Numbers of Existing DDMs for a Database	21
SYSMAIN Utility - Overview	22
SYSMAIN Utility - Overview	22
SYSMAIN General Information	23
SYSMAIN General Information	23
Objects	23
Environments	23
Functions	24
Invoking SYSMAIN	25
Invoking SYSMAIN	25
Invoking SYSMAIN Online	25
Terminating SYSMAIN Online	25
Invoking SYSMAIN in Batch Mode	25
Invoking SYSMAIN by a Subprogram	26
SYSMAIN Functions and Function Processing	27
SYSMAIN Functions and Function Processing	27
Commands	27
Function Processing	29

Command Line	31
Selection List	31
Enhanced Selection Criteria	34
SYSMAIN Direct Commands	35
SYSMAIN Direct Commands	35
General Direct Command Syntax	35
where-clause	35
with-clause	36
Sequence of Syntax Elements	36
Effects of Direct Commands	36
SYSMAIN Parameters and Keywords	37
SYSMAIN Parameters and Keywords	37
Description of Parameters and Keywords	38
CRITERIA (A1)	38
OBJECT TYPE (A15)	38
DATE FROM (A10)	38
TIME FROM (A5)	38
DATE TO (A10)	38
TIME TO (A5)	39
USER ID (A8)	39
TERMINAL ID (A8)	39
DDM DBID (N5)	39
DDM FNR (N5)	39
DDM NAME (A32)	39
ENVIRONMENT NAME (A8)	39
ERROR NUMBER FROM (N4)	40
ERROR NUMBER TO (N4)	40
ERROR TYPE (A1)	40
FDIC (A27)	40
FSEC (A27)	40
NEW NAME (A8)	41
NEW NUMBER FROM (N4)	41
NEW NUMBER TO (N4)	41
OBJECT NAME (A9)	41
OBJECT TYPE (A15)	42
PROFILE NAME (A8)	43
PROFILE TYPE (A3)	43
REPLACE (A1)	43
RULE NAME (A32)	43
RULE TYPE (A2)	44
SELECTION LIST (A1)	44
SET NUMBER (N2)	44
SET USER (A8)	44
SOURCE CIPHER (A8)	45
SOURCE DATABASE (N5)	45
SOURCE FILE (N5)	45
SOURCE LANGUAGE (A9)	45
SOURCE LIBRARY (A8)	45
SOURCE NAME (A8)	45
SOURCE PASSWORD (A8)	46
SUBFILE NAME (A8)	46
SUBFILE TYPE (A3)	46
TARGET CIPHER (A8)	46
TARGET DATABASE (N5)	46
TARGET FILE (N5)	46
TARGET LANGUAGE (A9)	46

TARGET LIBRARY (A8)	47
TARGET NAME (A8)	47
TARGET PASSWORD (A8)	47
XREF (A1)	47
Additional Keywords for Direct Commands	48
Range Notation for SYSMAIN Parameters	49
SYSMAIN Programming Objects	50
SYSMAIN Programming Objects	50
Programming Object Menus	50
Processing Status	54
Options H, I and L	54
Options B, X and Z	54
Options A, C and S	54
Direct Commands for Programming Objects	56
COPY and MOVE Direct Command Syntax	56
DELETE Direct Command Syntax	57
FIND, LIST and LISTLIB Direct Command Syntax	58
RENAME Direct Command Syntax	58
Programming Objects in Batch Mode	59
XREF Considerations for Programming Objects	59
XREF set to N	59
XREF set to Y or F	59
XREF set to F	59
XREF set to S	59
Errors	60
SYSMAIN Debug Environments	61
SYSMAIN Debug Environments	61
Debug Environment Menus	61
Processing Status	62
Direct Commands for Debug Environments	63
COPY and MOVE Direct Command Syntax	63
DELETE Direct Command Syntax	63
LIST Direct Command Syntax	64
RENAME Direct Command Syntax	64
Debug Environments in Batch Mode	64
SYSMAIN Error Messages	65
SYSMAIN Error Messages	65
Error Message Menus	65
Language Parameter Considerations	66
Renumber an Error Message	68
Processing Status	69
Option L	69
Options A, E and S	69
Direct Commands for Error Messages	70
COPY and MOVE Direct Command Syntax	70
DELETE Direct Command Syntax	71
FIND Direct Command Syntax	71
LIST Direct Command Syntax	71
RENAME Direct Command Syntax	72
Error Messages in Batch Mode	72
SYSMAIN Profiles	73
SYSMAIN Profiles	73
Profile Menus	73
Processing Status	75
Direct Commands for Profiles	75
COPY and MOVE Direct Command Syntax	75

DELETE Direct Command Syntax	76
LIST Direct Command Syntax	76
RENAME Direct Command Syntax	76
Profiles in Batch Mode	76
SYSMAIN Rules	77
SYSMAIN Rules	77
Rule Menus	77
Processing Status	79
Direct Commands for Rules	80
COPY and MOVE Direct Command Syntax	80
DELETE Direct Command Syntax	81
LIST Direct Command Syntax	81
Rules in Batch Mode	81
SYSMAIN DDMs	82
SYSMAIN DDMs	82
DDM Menus	82
Processing Status	83
Direct Commands for DDMs	83
COPY and MOVE Direct Command Syntax	84
DELETE Direct Command Syntax	84
LIST Direct Command Syntax	85
DDMs in Batch Mode	85
SYSMAIN DL/I Subfiles	86
SYSMAIN DL/I Subfiles	86
General Information	86
Processing Status	87
Direct Commands for DL/I Subfiles	87
COPY and MOVE Direct Command Syntax	87
DELETE and LIST Direct Command Syntax	88
Commands Issued to SYSMAIN	89
Commands Issued to SYSMAIN	89
SYSMAIN PF Keys	91
SYSMAIN PF Keys	91
SYSMAIN - Data Rejected	92
SYSMAIN - Data Rejected	92
Object Rejection and Reasons	92
SYSMAIN Error Notification	92
Data Entry Errors	93
Processing Errors	93
Special Considerations for Administrators	96
Special Considerations for Administrators	96
SYSMAIN Security	96
File Security	96
Natural Security	98
User Exits	99
MAINEX01 - First User Exit for Object Interrogation	100
MAINEX02 - Second User Exit for Object Interrogation	100
MAINEX03 - User Exit for Request Interrogation	101
MAINEX04 - User Exit for Modification of File Assignments	101
MAINEX05 - User Exit for Verification of Direct Commands	101
MAINEX06 - User Exit for SYSMAIN Initialization	101
MAINEX07 - User Exit for SYSMAIN Termination	102
MAINEX08 - User Exit for Nothing Found in Batch Mode	102
MAINEX09 - User Exit for Abnormal Termination in Batch Mode	102
MAINEX10 - User Exit for Command Errors in Batch Mode	103
MAINEX11 - User Exit for Setting Special Flags to SYSMAIN	103

SYSPARM Utility	104
SYSPARM Utility	104
Invoking SYSPARM	104
List Profiles	105
Display Profile	105
Add New Profile	105
Specifying Parameters	106
Functions	106
Help for Parameter	106
Modify Profile	107
Copy Profile	107
Delete Profile	107
Debugging and Monitoring - Overview	108
Debugging and Monitoring - Overview	108
Natural Debugger - Overview	109
Natural Debugger - Overview	109
Concepts of the Natural Debugger	110
Concepts of the Natural Debugger	110
Debug Entries/Spies	110
Debug Window	111
Start the Natural Debugger	112
Start the Natural Debugger	112
Invoke the Natural Debugger	112
Default Object	112
Set Test Mode ON/OFF	113
Set Test Mode ON/OFF	113
Debug Environment Maintenance	114
Debug Environment Maintenance	114
Load Debug Environment	114
Save Debug Environment	114
Reset Debug Environment	115
Delete Debug Environment	115
Spy Maintenance	116
Spy Maintenance	116
Activate Spy	116
Deactivate Spy	116
Delete Spy	116
Display Spy	117
Modify Spy	117
Breakpoint Maintenance	118
Breakpoint Maintenance	118
Conditions of Use	118
Activate Breakpoint	119
Deactivate Breakpoint	119
Delete Breakpoint	119
Display Breakpoint	119
Modify Breakpoint	119
Display/Modify Breakpoint Screen	120
Set Breakpoint	121
Watchpoint Maintenance	123
Watchpoint Maintenance	123
Invoke Watchpoint Maintenance	123
Activate Watchpoint	123
Deactivate Watchpoint	124
Delete Watchpoint	124
Display Watchpoint	124

Modify Watchpoint	125
Display/Modify Watchpoint Screen	125
Set Watchpoint	126
Watchpoint Operators	127
Call Statistics	128
Call Statistics	128
Set Call Statistics ON/OFF	128
Display All Objects	128
Display Called Objects	129
Display Non-Called Objects	129
Print Objects	129
Print Options	130
Statement Execution Statistics	131
Statement Execution Statistics	131
Set Statement Execution Statistics ON/OFF/COUNT	131
Setup Options	131
Activate/deactivate Statistics	132
Invoke Statement Execution Statistics	132
Delete Statement Execution Statistics	133
Display Statement Execution Statistics	133
Display All Statement Lines	134
Display Executed Statement Lines	135
Display Non-Executed Statement Lines	135
Print Statements	135
Print Options	135
Variable Maintenance	136
Variable Maintenance	136
Display Variable	136
Modify Variable	137
List Object Source	138
List Object Source	138
Maintain Breakpoints	138
Execution Control Commands	140
Execution Control Commands	140
ESCAPE BOTTOM	140
ESCAPE ROUTINE	140
EXIT	140
GO	141
NEXT	141
RUN	141
STEP	141
STEP SKIPSUBLEVEL	141
STOP	141
Navigation and Information Commands	142
Navigation and Information Commands	142
BREAK	142
FLIP	142
LAST	142
OBJCHAIN	142
ON/OFF	142
PROFILE	143
Edit Profile Screen	143
SCAN	143
SCREEN	143
SET OBJECT	144
STACK	144

SYSVARS	144
TEST ON/OFF	144
Command Summary and Syntax	145
Command Summary and Syntax	145
All Debug Commands	145
Syntax Diagrams	149
ACTIVATE	150
DEACTIVATE	150
DELETE	150
DISPLAY	151
LIST	151
LOAD	152
MODIFY	152
PRINT	152
RESET	153
SAVE	153
SET	153
DBLOG Utility - Logging Database Calls	154
DBLOG Utility - Logging Database Calls	154
Executing DBLOG	155
Executing DBLOG	155
Data Processing and Storage	155
Activating and Deactivating DBLOG	155
Using Selective DBLOG	156
DBLOG Menu	157
DBLOG Menu	157
DBLOG Menu - Functions	157
DBLOG Menu - Specifying Restrictions	158
Specifying Buffers	159
DBLOG Trace Screen	160
DBLOG Trace Screen	160
DBLOG Trace Screen - Adabas Commands	160
Invoking DBLOG Trace - Adabas Commands	160
DBLOG Trace Adabas Commands - Fields and Functions	160
Displaying Buffers on the DBLOG Trace Screen	161
DBLOG Trace Screen - DL/I Calls	163
Invoking DBLOG Trace - DL/L Calls	163
DBLOG Trace DL/I Calls - Fields and Functions	164
DBLOG Trace Screen - SQL Statements	164
Invoking DBLOG Trace - SQL Statements	164
DBLOG Trace SQL Statements - Fields and Functions	166
DBLOG - Snapshot Function	168
DBLOG - Snapshot Function	168
Snapshot Function - Adabas Commands	168
Invoking Snapshot - Adabas Commands	168
Displaying Buffers on the Snapshot Report	169
Snapshot Function - DL/I Calls	169
Invoking Snapshot - DL/I Calls	170
Snapshot Report Information - DL/I Calls	170
Snapshot Function - SQL Statements	171
Invoking Snapshot - SQL Statements	171
Snapshot Report Information - SQL Statements	172
DBLOG Direct Commands	173
DBLOG Direct Commands	173
Syntax Diagrams	173
Parameters	173

ADACALL - Issuing Adabas Direct Calls	175
ADACALL - Issuing Adabas Direct Calls	175
Invoking ADACALL	175
ADACALL Parameters	176
Adabas OP Command	177
ADACALL Commands and PF Keys	177
User Exit ADAEXIT	179
SYSBPM Utility - Buffer Pool Management	180
SYSBPM Utility - Buffer Pool Management	180
Invoking and Operating SYSBPM	181
Invoking and Operating SYSBPM	181
Invoking SYSBPM	181
SYSBPM Main Menu - Fields and Functions	182
Maintenance of Further Buffer Pools	183
Display Buffer Pools	184
Select Buffer Pool	184
Reset Buffer Pool	184
SYSBPM in a Sysplex Environment	184
SYSBPM - Buffer Pool Statistics	185
SYSBPM - Buffer Pool Statistics	185
General Buffer Pool Statistics	185
Buffer Pool Load/Locate Statistics	187
Buffer Pool Fragmentation	189
Internal Function Usage	190
Buffer Pool Hash Table Statistics	190
SYSBPM - BP Cache Statistics	192
SYSBPM - BP Cache Statistics	192
General BP Cache Statistics	192
BP Cache Call Statistics	193
BP Cache Hash Table Statistics	195
SYSBPM - Individual Object Statistics	196
SYSBPM - Individual Object Statistics	196
Columns	196
Navigation	198
Line Commands	198
SYSBPM - Object Directory Information	200
SYSBPM - Object Directory Information	200
Fields	200
Functions	201
SYSBPM - Display Object Hexadecimal	203
SYSBPM - Display Object Hexadecimal	203
Navigation	203
SYSBPM - Delete Object from Buffer Pool	204
SYSBPM - Delete Object from Buffer Pool	204
SYSBPM - Blacklist Maintenance	205
SYSBPM - Blacklist Maintenance	205
Maintain Blacklist	205
Adding Objects	206
Modifying Objects	206
Deleting Objects	206
List Object Sets	207
Edit Object Set	207
Creating Object Sets	207
Modifying Object Sets	208
Add Object Set to Blacklist	209
Delete Object Set from Blacklist	209

Delete Object Set Text Member	209
BPMBLBAT - Blacklist Maintenance in Batch Mode	210
SYSBPM - Preload List Maintenance	212
SYSBPM - Preload List Maintenance	212
List Preload Lists	212
Edit Preload List	212
Creating Preload Lists	213
Modifying Preload Lists	213
Generate Preload List from Buffer Pool	214
Delete Preload List	215
SYSBPM Direct Commands	216
SYSBPM Direct Commands	216
SYSEDT Utility - Editor Buffer Pool Services	220
SYSEDT Utility - Editor Buffer Pool Services	220
Defining a Natural Security Library Profile	220
Invoking and Operating SYSEDT	220
Invoking the SYSEDT Utility	220
Invoking a SYSEDT Function	221
Using Direct Commands Help	221
General Information	221
Generation Parameters	222
Modifying Parameter Values	223
Users	224
Logical Files	224
Recovery Files	225
System Administration Facilities	226
SYSRDC Utility	227
SYSRDC Utility	227
Overview	228
Data-Collecting Events	228
Collected Data	229
Activating SYSRDC	231
Exit Points for External Monitoring/Accounting Programs	232
Return Codes	233
Sample Exit Programs	233
Trace Recording - CALL Interface	234
Program CMRDC	234
CMRDC Functions	235
CMRDC Return Codes	238
Sample Programs in Library SYSRDC	238
SYSTP Utility - Overview	239
SYSTP Utility - Overview	239
General SYSTP Functions	240
General SYSTP Functions	240
Natural Monitoring - SYSMON	240
Activate/Deactivate Monitor	240
Display Monitor Terminal Statistics	241
Display Monitor Program Statistics	243
Natural Print/Work Files - SYSFILE	244
Natural Swap Information	245
Administration	246
Debugging Facilities	246
Information	247
Parameter Service	249
Status Information	249
Buffer Usage Statistics - BUS	249

Individual Buffers	252
Natural Subsystems and Roll Server Information	255
Natural Thread Usage Statistics	255
Show Physical GETMAIN Statistics	257
SYSTP Functions under CICS	258
SYSTP Functions under CICS	258
Natural User Sessions	259
User Session Statistics	260
Natural Roll Facilities	264
Natural Thread Groups	265
Natural Storage Threads	266
NCI Global System Information	267
NCI Generation Options	268
Natural Thread Group Definitions	268
Own Natural User Session	269
CICS Task Information	269
System Administration Facilities	269
System Snapshot for Logging	269
Reset System Highwater Marks	269
Applied NCI ZAPs	269
SYSTP Functions under IMS/TM	270
SYSTP Functions under IMS/TM	270
Broadcasting	270
Display Environment Data	270
Monitoring	271
Multi Session	271
Applied NII ZAPs	271
SYSTP Functions under TIAM and UTM	272
SYSTP Functions under TIAM and UTM	272
P-Key Utility	272
Load User Values	273
Set Key Assignment Mode	274
Loading Send-Key Codes to P Keys	275
User Exit LPFSUP01	276
Show Common Memory Pools	276
SYSTP in Batch Mode	277
SYSTP in Batch Mode	277
Invoking SYSTP in Batch Mode	277
Evaluating the Log File	277
NATPAGE Utility - Screen Paging	279
NATPAGE Utility - Screen Paging	279
Natural Recording Utility	280
Natural Recording Utility	280
Purpose of Recording	280
Data and Functions Recorded	280
Recording a Session	281
Specifying Libraries	281
Activating a Recording	281
Deactivating a Recording	281
Playing Back a Recording	282
Step Mode and Background Mode	282
Activating a Playback	282
Interrupting a Playback	283
Manipulating a Recording	283

Natural Utilities for Mainframes - Overview

This documentation describes the functions of the Natural utilities available in mainframe environments:

- NATRJE Submits JCL cards from a Natural program to the operating system for scheduling and execution.
- Debugging and Monitoring Debugging, testing and monitoring Natural objects and applications.
- SYSDDM Creates and maintains DDMs.
- SYSMAIN Transfers Natural programming objects, error messages, DDMs and other objects from one library to another.
- SYSPARM Creates and maintains Natural parameter profiles.

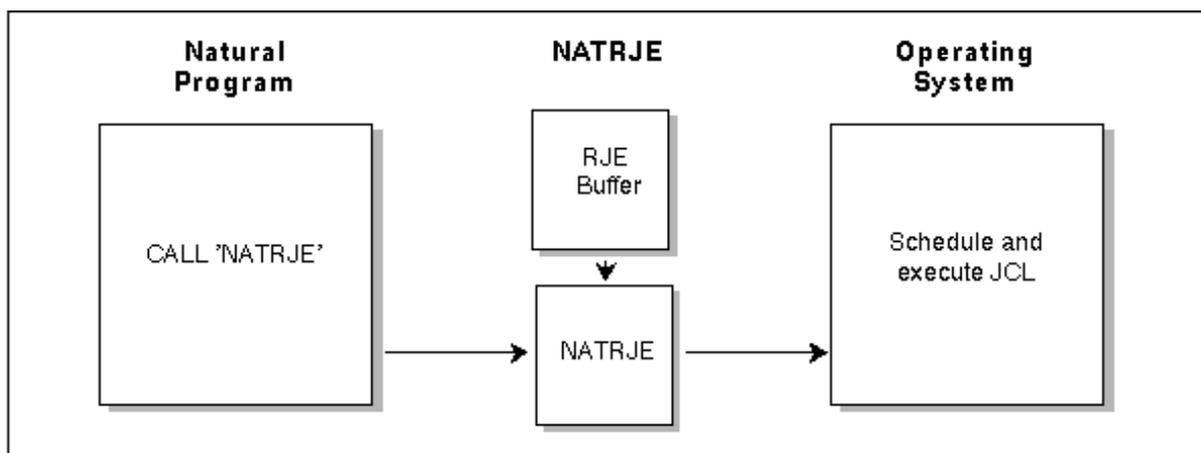
NATRJE - Natural Remote Job Entry

The NATRJE utility (Natural Remote Job Entry) can be used to submit JCL cards from a Natural program to the operating system for scheduling and execution. For example, it is possible to start a Natural batch job with NATRJE.

This section covers the following topics:

- NATRJE General Information
- Calling NATRJE from a Natural Program
- NATRJE Return Codes
- NATRJE User Exit
- NATRJE Features Applicable to UTM/TIAM

NATRJE General Information



1. The Natural program calls the NATRJE utility for the purpose of submitting JCL cards to be executed by the operating system.
2. NATRJE collects the JCL cards into the RJE buffer until the Natural program indicates that the job is complete. The RJE buffer holds the JCL cards before they are submitted. The initial size of the RJE buffer is determined by the RJESIZE profile parameter (as described in the Natural Parameter Reference documentation). If a given job does not fit into the RJE buffer, the buffer is automatically enlarged. The maximum size of a job is determined by the thread or region size.
3. NATRJE transfers the JCL cards to the operating system internal job queue for scheduling and execution by the operating system.

Note for BS2000/OSD:

In BS2000/OSD environments, when generation of the job is completed, NATRJE transfers the JCL cards to a BS2000/OSD dataset, which is generated by NATRJE. The dataset is a SAM file and is submitted via ENTER to the BS2000/OSD operating system.

Calling NATRJE from a Natural Program

Below is information on:

- Invoking NATRJE
- Example Programs

Invoking NATRJE

 **To invoke the NATRJE utility**

- Use a CALL statement in the Natural program.

The CALL statement has the following syntax:

```
CALL 'NATRJE' parm1 parm2 parm3 parm4
```

The parameters specified in the CALL statement are:

Parameter	Explanation
<i>parm1</i>	The starting JCL card of the table which contains one or more 80-character JCL cards to be submitted.
<i>parm2</i>	A 4-byte binary field which contains the number of 80-character JCL cards to be submitted.
<i>parm3</i>	<p>A 1-byte alphanumeric field used to indicate if all JCL cards have been submitted:</p> <ul style="list-style-type: none"> ' ' Not the last call for the current job. A further JCL card follows with the next CALL statement. The JCL cards are collected into the RJE buffer. B BS2000/OSD and OS/390 only: Last call for the current job. Under BS2000/OSD: The job is generated, written to the dataset, but not started automatically. Under OS/390 (batch and TSO, IMS/TM and CICS): The job is written to the internal reader dataset but not submitted. If function L is called subsequently, the internal reader is closed and the job(s) is submittedand. In addition, the internal reader is closed and the job is submitted: <ul style="list-style-type: none"> on a screen I/O (IMS/TM), or on session termination (OS/390 batch, TSO and IMS/TM). C Flush the current job. The job is not submitted to the system. (Under BS2000/OSD, no dataset is created.) L Last call for the current job. The job is submitted to the system. <p>BS2000/OSD environments: see Additional Values for the <i>parm3</i> Parameter.</p>
<i>parm4</i>	A 2-byte binary field in which NATRJE returns a response code.

Example Programs

Below are Natural example programs for use under the operating systems:

- OS/390
- VSE/ESA
- BS2000/OSD (Example 1 and Example 2)

Example Program - OS/390:

The following is a Natural example program that submits, in one call to NATRJE, a three-card JCL stream.

```

DEFINE DATA LOCAL
01 COUNT (B4)
01 FLAG (A1)
01 RETHEX (B2)
01 CARDS (A240)
01 REDEFINE CARDS
  02 CARD1 (A80)
  02 CARD2 (A80)
  02 CARD3 (A80)
END-DEFINE
MOVE '//JOB JOB CLASS=G,MSGCLASS=X' TO CARD1
MOVE '//XXX EXEC PGM=IEFBR14' TO CARD2
MOVE '//DD1 DD DSN=NATRJE.SOURCE,DISP=SHR' TO CARD3
MOVE 3 TO COUNT
MOVE 'L' TO FLAG
CALL 'NATRJE' CARDS COUNT FLAG RETHEX
IF RETHEX = H'0000'
  WRITE 'JOB submitted successfully'
ELSE
  WRITE 'ERROR from NATRJE' RETHEX
END-IF
END

```

Example Program - VSE/ESA:

The following is a Natural example program that submits, in three calls to NATRJE, a seven-card JCL stream.

```

DEFINE DATA LOCAL
01 COUNT (B4)
01 FLAG (A1)
01 RETHEX (B2)
01 CARDS (A240)
01 REDEFINE CARDS
  02 CARD1 (A80)
  02 CARD2 (A80)
  02 CARD3 (A80)
END-DEFINE
MOVE '* $$ JOB JNM=DSERV,CLASS=0,DISP=D' TO CARD1
MOVE '* $$ LST CLASS=A,DISP=D' TO CARD2
MOVE '// JOB DSERV TO DSERV SOURCE MEMBERS' TO CARD3
MOVE 3 TO COUNT
CALL 'NATRJE' CARDS COUNT FLAG RETHEX
PERFORM RETCODE-CHECK
MOVE '// EXEC PROC=NATSPLP' TO CARD1
MOVE '// EXEC DSERV' TO CARD2
MOVE ' DSPLYS SD' TO CARD3
MOVE 3 TO COUNT
CALL 'NATRJE' CARDS COUNT FLAG RETHEX
PERFORM RETCODE-CHECK
MOVE '/*' TO CARD1
MOVE '/&' TO CARD2
MOVE '* $$ EOJ' TO CARD3
MOVE 3 TO COUNT
MOVE 'L' TO FLAG

```

```

CALL 'NATRJE' CARDS COUNT FLAG RETHEX
DEFINE SUBROUTINE RETCODE-CHECK
IF RETHEX NE H'0000'
  WRITE 'ERROR from NATRJE:' RETHEX
STOP
END-IF
END-SUBROUTINE
END

```

Example Program 1 - BS2000/OSD:

The following is a Natural example program that submits, in three calls to NATRJE, a nine-card JCL stream.

```

DEFINE DATA LOCAL
  01 COUNT (B4)
  01 FLAG (A1)
  01 RETHEX (B2)
  01 CARDS (A240)
  01 REDEFINE CARDS
    02 CARD1 (A80)
    02 CARD2 (A80)
    02 CARD3 (A80)
END-DEFINE
MOVE '/LOGON' TO CARD1
MOVE '/SYSFILE SYSDTA=(SYSCMD)' TO CARD2
MOVE '/SYSFILE SYSIPT =IPT.PARM' TO CARD 3
MOVE 3 TO COUNT
  CALL 'NATRJE' CARDS COUNT FLAG RETHEX
  IF RETHEX NE H'0000' DO
    WRITE RETHEX (EM=HH)
  END-IF
MOVE '/SETSW ON=2' TO CARD1
MOVE '/EXEC NATB21' TO CARD2
MOVE 'LOGON APPLIC' TO CARD3
MOVE 3 TO COUNT
  CALL 'NATRJE' CARDS COUNT FLAG RETHEX
  IF RETHEX NE H'000' DO
    ...
    ...
  END-IF
MOVE 'RUNPGM' TO CARD1
MOVE 'FIN' TO CARD2
MOVE '/LOGOFF' TO CARD3
MOVE 3 TO COUNT
MOVE 'L' TO FLAG
  CALL 'NATRJE' CARDS COUNT FLAG RETHEX
  ...
  ...
  ...
END

```

Example Program 2 - BS2000/OSD:

The following is a Natural example program that submits, in one call to NATRJE, a nine-card JCL stream.

```

DEFINE DATA LOCAL
  01 COUNT (B4)
  01 FLAG (A1)
  01 RETHEX (B2)
  01 CARD1 (A80)
  01 CARD2 (A80)
  01 CARD3 (A80)

```

```

01 CARD4 (A80)
...
01 CARD9 (A80)
END-DEFINE
MOVE '/LOGON' TO CARD1
MOVE '/SYSFILE SYSDTA=(SYSCMD)' TO CARD2
...
MOVE '/LOGOFF' TO CARD9
MOVE 9 TO COUNT
MOVE 'L' TO FLAG
CALL 'NATRJE' CARD1 COUNT FLAG RETHEX
...
END
    
```

NATRJE Return Codes

A CALL to the module NATRJE results in one of the following return codes being returned in the fourth parameter of the CALL statement. There are return codes that apply to all environments and additional codes that are dependent on the operating system:

- Return Codes Common to all Environments
- Additional Return Codes for VSE/ESA
- Additional Return Codes for BS2000/OSD

Return Codes Common to all Environments

Hexadecimal	Decimal	Meaning
00	00	Normal Return
04	04	RJE utility not available
08	08	RJE utility disabled; a possible reason is that the RJESIZE parameter is set to 0
0C	12	Invalid Number of JCL cards
10	16	Invalid Function Code
14	20	No RJE Buffer Space available
18	24	Invalid Number of Parameters
1C	28	I/O Error during Submit
20	32	Job flushed by user exit NREXPG (see also NATRJE User Exit below)

Additional Return Codes for VSE/ESA

01xx	PUTSPOOL Error, xx is R15 Contents
02xx	PUTSPOOL Error, xx is XECB+4
03xx	XECBTAB Define Error, xx is Return Code
04xx	XECBTAB Delete Error, xx is Return Code

Additional Return Codes for BS2000/OSD

9001	No RJE buffer found
9002	No buffer space available
9003	Missing LOGON command
9004	Only LOGON cards generated
9005	Too many LOGON parameters
D010	Error in ENTER macro
Dxxx	Operating system error: The error message is sent directly to the user program; the BS2000/OSD HELP command provides additional information.

NATRJE User Exit

A user exit capability for Natural Remote Job Entry is provided. After the job is complete, each JCL card is passed to the exit before it is submitted to the operating system. The following data are available to the exit:

- the JCL card to be submitted,
- a return code field,
- the name of the Natural program currently being executed,
- the Natural user identification,
- a 240-byte work area.

After each call, the exit passes a return code to NATRJE indicating one of the following events:

Code	Explanation
0	Submission: the card is submitted; the exit may modify the card before submission.
4	Termination: the card is submitted; the exit is disabled for further cards of the current job.
8	Insertion: the card is skipped (based on the assumption that it contains only an INSERT character); additional specified cards are submitted.
10	Deletion: the card is not submitted.
12	The current job is flushed.

An example of the user exit, called NREXPG, is available as member XNATRJE in the Natural source library. The exit can be assembled and linked according to the rules of programs specified as CSTATIC. However, a CSTATIC entry for NREXPG is not required.

NATRJE Features Applicable to UTM/TIAM

Below is information on:

- Additional Values for the parm3 Parameter
- Name of BS2000/OSD Dataset

Additional Values for the parm3 Parameter

Value	Explanation
A	Combination of values T and E.
E	The job is generated and completed. Before submission to the BS2000/OSD operating system, the parameter ERASE=YES is added to the ENTER parameter.
T	The job is generated and completed. Before submission to the BS2000/OSD operating system, a time limit is calculated using the Natural MT parameter. If MT is set to 0, the time limit is generated as NTL. The calculated time limit is added to the ENTER parameter via the TIME= operand.

When using the values T, E or A, NATRJE does not check whether the parameters TIME= or ERASE= exist in the user-created LOGON cards.

Name of BS2000/OSD Dataset

The name of the BS2000/OSD dataset created by NATRJE for the JCL cards is as follows:

E.DDMMYY.HHMMSSSS.*program-name*.*user-id*

Parameter	Specifies
DD	The day of the dataset creation.
MM	The month of the dataset creation.
YY	The year of the dataset creation.
HH	The hour of the dataset creation.
MM	The minute of the dataset creation.
SSSS	The seconds and milliseconds of the dataset creation.
<i>program-name</i>	The name of the Natural program that creates the dataset.
<i>user-id</i>	The corresponding Natural user ID.

SYSDDM Utility

The utility SYSDDM is used to create and maintain Natural data definition modules (DDMs).

- DDMs
 - SYSDDM and Predict
 - Invoking SYSDDM
 - Generate DDM from Adabas FDT
 - Catalog DDM
 - Edit DDM
 - Delete DDM
 - List DDMs
 - List DDMs with Additional Information
 - Copy DDM to Another FDIC File
 - Show Defined DBIDs and Used FNRs
-

DDMs

For general information on DDMs, refer to the Natural Programming Guide.

A Natural application can only access a database file if a corresponding DDM has been created and cataloged for the file.

Cataloged DDMs are stored in the Natural system file FDIC.

A DDM can be created either with the SYSDDM utility (as described in this section) or with Predict (as described in the Predict documentation).

SYSDDM and Predict

If Predict is installed at your site, you should **not** use SYSDDM; instead, it is recommended that you use the functions offered by Predict for the creation and maintenance of DDMs.

With Predict, it is possible to control the availability of SYSDDM. It may therefore be that the use of SYSDDM has been restricted and certain SYSDDM functions are not available to you. Please see the Predict documentation for further information.

Invoking SYSDDM

▶ To invoke the SYSDDM utility

- Enter the SYSDDM system command.
The main menu of SYSDDM is displayed:

```

09:18:45          ***** NATURAL SYSDDM UTILITY *****          1999-12-02
User SAG          - Menu -          FDIC (10,160)
Code  Function          Work area empty
G      Generate DDM from ADABAS FDT
C      Catalog DDM
E      Edit DDM
U      Delete DDM
L      List DDMs
X      List DDMs with Additional Information
M      Copy DDM to Another FDIC File
S      Show Defined DBIDs and Used FNRs
B      SQL Services
D      DL/I Services
.      Exit

Code ..... _          FDIC Type ..... A
DDM Name .. _____ DDM Type ..... _
FNR ..... 0          DBID .. 0          ADABAS Password ..
Replace ... N          DBID Type ..... 6

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

Overview of Functions

From the SYSDDM main menu you can select the following functions:

Function	Explanation
Generate DDM from Adabas FDT	Generates a DDM from an Adabas field description table (FDT) and places it in the SYSDDM work area for further processing.
Catalog DDM	The DDM currently in the SYSDDM work area is cataloged, making it available for use within Natural applications. The DDM must have been placed in the work area by the function Generate DDM from Adabas FDT, or have been entered by using the function Edit DDM. For a VSAM DDM (DDM Type = V), SYSDDM prompts you for additional information; for details, see the Natural for VSAM documentation, Natural File Access.
Edit DDM	Reads a DDM from the system file FDIC and into the SYSDDM work area, where it can be edited.
Delete DDM	Deletes a cataloged DDM from the system file FDIC.
Copy DDM	Copies a DDM from one FDIC system file to another.
List DDMs	Displays a list of the DDMs stored in the specified FDIC system file. From the list, you can select individual DDMs for further processing. This function corresponds to the system command LIST DDM (see LIST in the Natural Command Reference documentation).
List DDMs with Additional Information	Displays a list of the DDMs stored in the specified FDIC system file. From the list, you can select individual DDMs for further processing. This function differs from the List DDMs function in that it displays additional items of information on the individual DDMs.
Show Defined DBIDs and Used FNRs	This function shows you which DBIDs are defined, as well as all file numbers of a given DBID for which DDMs have been defined.
SQL Services	This function is available only if Natural for DB2 or Natural for SQL/DS is installed. It is used to generate DDMs from DB2 or SQL/DS tables and is described in the documentation Natural for DB2 and Natural for SQL/DS respectively.
DL/I Services	This function is available only if Natural for DL/I is installed. It is used to maintain the Natural for DL/I environment. Functions are provided for inquiry into and modification of structures, such as DL/I Database Descriptions (DBDs), Program Specification Blocks (PSBs), Program Communication Blocks (PCBs), DDMs and segment layouts. This function is described in the documentation Natural for DL/I.

The following parameters can be specified on the SYSDDM main menu for the various functions:

Parameter	Explanation
DDM Name	The name of the DDM to be processed. To process multiple DDMs, you can use asterisk notation for the name.
FNR	The file number of the database file for which the DDM is (to be) defined.
DBID	The database which contains the file for which the DDM is (to be) defined.
Replace	Y A DDM which is being copied or cataloged will replace an existing DDM of the same name. N Existing DDMs are not replaced.
FDIC Type (display only)	The database type of the system file. Possible types are the same as for DDM Type (see below).
DDM Type	The type of DDM. Possible types are: A Adabas V VSAM 2 DB2 D DL/I P Entire System Server S SAP C Command processor
Adabas Password	The password required by Adabas if Adabas Security is installed.
DBID Type (display only)	The database type of the database specified in the DBID field. Possible types are the same as for DDM Type (see above); exception: for an Adabas database, the Adabas version (5 or 6) is displayed.

Generate DDM from Adabas FDT

This function is used to generate a DDM from an existing Adabas Field Description Table (FDT).

You have to enter the file number (FNR) of the Adabas file.

You can also enter a DBID. If you do not enter one, the DBID currently in effect for the session is used.

The generated DDM is placed in the SYSDDM work area for further processing.

Catalog DDM

To catalog a DDM, you either select the function Catalog DDM on the SYSDDM main menu or enter the command CATALOG in the command line of the DDM editor.

For this function, you have to specify the DDM name and file number (FNR).

The DBID, if not entered, is generated dynamically at execution time based on the DBID of the Natural user system file(FUSER) in use (see also the UDB profile parameter in the Natural Parameter Reference documentation).

For additional options for VSAM files, see the documentation Natural for VSAM.

Edit DDM

When you modify a DDM, all programming objects which reference this DDM must be cataloged again.

This function reads a DDM from the system file FDIC and places it into the SYSDDM work area, where you can edit it.

If you invoke this function without specifying a DDM, and there is no DDM already in the work area, an empty work area is displayed, allowing you to manually enter a DDM definition.

Instead of entering a complete DDM manually, you can read an existing DDM into the work area, modify it, and catalog it under a different name.

Below is information on:

- DDM Editor
- Field Attributes
- DDM Editor Commands
- Extended Field Editing

DDM Editor

```

09:26:50          ***** EDIT DDM (ADA) *****          1999-12-02
DDM Name EMPLOYEES          Def.Seq.          DBID          0 FNR          316
Command
I T L DB Name          F Leng  S D Remark
----- top -----
  1 AA PERSONNEL-ID          A 8.0          D
*          C=NNNNNNNN
*          C=COUNTRY
G 1 AB FULL-NAME
  2 AC FIRST-NAME          A 20.0  N
  2 AD MIDDLE-I          A 1.0    N
  2 AE NAME          A 20.0          D
  1 AD MIDDLE-NAME          A 20.0  N
  1 AF MAR-STAT          A 1.0    F
*          M=MARRIED
*          S=SINGLE
*          D=DIVORCED
*          W=WIDOWED
  1 AG SEX          A 1.0    F
  1 AH BIRTH          D 6.0          D
  1 AH N)BIRTH          I 2.0          D
G 1 A1 FULL-ADDRESS

SYSDDM 4393: DDM read into the source area.

```

If you enter the command **HELP** (or a question mark) in the command line, the editor help information is displayed.

The header of the DDM editor contains the following information:

DDM Name	The name used to reference the DDM in a Natural program. The name must be unique within the specified Natural system file.
Def. Seq.	The default sequence by which the file is read when it is accessed with a READ LOGICAL statement in a Natural program.
DBID	The database in which the file to be accessed with the DDM is contained. If 0 (zero) is specified, the default DBID for the Natural user system file (FUSER) as defined in the Natural parameter module is used.
FNR	The number of the file being referenced. If an Adabas file is used, the Adabas file number must be entered. If a DL/I segment type is used, the file number specified is used internally by Natural for DL/I. For VSAM files, see the Natural for VSAM documentation.

Field Attributes

The DDM itself comprises the following field definition attributes which can be entered or modified:

Attribute	Explanation
I	<p>Line indicator.</p> <p>This field is used by the DDM editor to mark lines.</p> <p>E Lines containing an error detected during execution of a CHECK command.</p> <p>S Lines containing a scanned value.</p> <p>X/Y Lines selected for copy/move operation.</p>
T	<p>Field Type:</p> <p>G Group header</p> <p>M Multiple-value field</p> <p>P Periodic group header</p> <p>* Comment line</p> <p><i>blank</i> Elementary field</p> <p>Note: Groups defined in a DDM need not necessarily be defined as groups in the Adabas FDT</p>
L	<p>Level number assigned to the field.</p> <p>Valid level numbers are 1 - 7.</p> <p>Level numbers should be specified in consecutive ascending order.</p>
DB	<p>For Adabas files, the Adabas two-character field name.</p> <p>For DL/I segment types, the 2-character code which is used in DL/I.</p> <p>For VSAM files, see the documentation Natural for VSAM.</p>
Name	<p>An external field name of 3 to 32 characters.</p> <p>This is the name used within Natural programs to reference the field.</p>
F	<p>Field format.</p> <p>For valid formats, refer to Definition of Format and Length in User-Defined Variables (General Information, Natural Programming Reference documentation).</p>
Leng	<p>Standard field length.</p> <p>This length can be overridden in a Natural program.</p> <p>For numeric fields (format N), the length is specified as <i>nn.m</i>, where <i>nn</i> represents the number of digits before the decimal point and <i>m</i> represents the number of digits after the decimal point.</p>

Attribute	Explanation
S	<p>Null-value suppression option (only for Adabas files):</p> <p>N Indicates that the field is defined with the Adabas null-value suppression option. This means that null values for the field are not stored in the inverted list and are not returned when the field is used in the WITH clause of a FIND statement, or in a HISTOGRAM or READ LOGICAL statement. If the Remarks column contains NC (not counted), an N in this column indicates that the field is defined with the SQL null-value option. Below this field, the corresponding null-indicator field is listed.</p> <p>M Indicates that the field is defined with the SQL null-value option "not null". The Remarks column for this field contains "NN NC" (not null, not counted). Below this field, the corresponding null-indicator field is listed.</p> <p>F Indicates that the field is defined with the Adabas fixed-storage option.</p>
D	<p>Descriptor Option.</p> <p>A Indicates that the field is an alternate index for a VSAM file.</p> <p>D Indicates that the field is an Adabas descriptor.</p> <p>H Indicates that the field is an Adabas hyperdescriptor. A hyperdescriptor is a user exit in Adabas and has to Natural the same functionality as a phonetic descriptor.</p> <p>N Indicates that the field is defined as a non-descriptor. A non-descriptor is not a descriptor, but can be used as a search field for a so-called non-descriptor search.</p> <p>P Indicates that the field is an Adabas phonetic descriptor.</p> <p>S Indicates that the field is an Adabas superdescriptor. If a superdescriptor contains a multiple-value field or a field from a periodic group (or part of such a field), the superdescriptor is marked with an M or a P in the Field Type column; this enables Natural to create the correct search algorithms for this superdescriptor. For a DL/I segment type, S indicates a superdescriptor; that is, a search field of a parent segment.</p> <p>U Indicates that the field is an Adabas subdescriptor. If a subdescriptor contains a multiple-value field or a field from a periodic group (or part of such a field), you have to mark the subdescriptor with an M in the Field Type column. This enables Natural to create the correct search algorithms for this subdescriptor.</p> <p>X Indicates an alternate subdescriptor or superdescriptor; that is, an alternate index for a VSAM file.</p> <p>For VSAM files, see the Natural for VSAM documentation.</p>
Remarks	A comment which applies to a field and/or the DDM.

DDM Editor Commands

Most of the editor and line commands available with the Natural program editor are also available in the DDM editor. Not available are special commands, such as PROFILE, RENUMBER, SET, SHIFT etc. and some line commands. Refer to the description of the program editor in the Natural User's Guide for Mainframes for more details on editor commands.

The following editor commands are also available:

CATALOG

```
CATALOG [DDM-name] [REPLACE]
```

Catalogs the DDM in the work area. If the DDM definition is already cataloged, the replace option must be used.

CHECK

```
CHECK
```

Validates the DDM in the work area against the Adabas FDT. Should any inconsistency occur, the field definition causing the error is marked for correction.

CLEAR

```
CLEAR
```

Clears the work area.

HELP

```
{HELP  
?}
```

Displays editor help information.

LENGTH / SIZE

```
{ LENGTH } [from-field to-field]
{ SIZE }
```

Calculates the maximum length for one record in bytes. If you specify *from-field* and *to-field*, the length from *from-field* to *to-field* is calculated only.

LIST DDM

```
LIST DDM [DDM-name]
```

Lists another DDM without leaving the DDM editor (corresponds to the system command LIST DDM).

READ

```
READ [DDM-name]
```

Reads a DDM into the work area. Any DDM currently in the work area is overwritten.

QUIT

```
{ QUIT }
```

Leaves the DDM editor. The DDM in the work area is still available until another DDM is read into the work area, the work area is used otherwise (for example, by the program editor) or the Natural session is terminated.

UNCAT

```
UNCAT [DDM-name]
```

Deletes either the DDM currently in the work area or an optionally specified DDM from the current library.

Extended Field Editing

With the DDM editor, you can also enter or modify DDMs at field level. You can specify default options for field headers and edit masks (as well as additional field definitions specific to VSAM files).

The extended editing mode is invoked by entering the line command **.E** in the first positions of the line containing the field. The Extended Field Editing screen is displayed:

```

09:50:05          ***** EDIT DDM (ADA) *****          1999-12-02
                   - Extended Field Editing -
DDM Name EMPLOYEES          Def.Seq.          DBID    0 FNR    316

I T L DB Name          F Leng  S D Remark
----- top -----
   1 AA PERSONNEL-ID          A 8.0    D
-----

Field Header ..... PERSONNEL/ID_____
Field Edit Mask .. _____
    
```

On this screen, you can specify field headers and edit masks to be applied when the field is used in a DISPLAY or INPUT statement, as well as further specifications for VSAM DDMs. All the other information specific to the field (field type, length, name, format, remarks) can also be modified on this screen.

When you press ENTER on this screen (with or without having entered anything), you will be returned to the DDM editor screen.

You can select a range of field definitions for editing by entering **.Ennn** where **nnn** is the number of fields to be selected.

For extended field editing in VSAM DDMs, see the Natural for VSAM documentation.

Delete DDM

This function is used to delete a single cataloged DDM from the system file FDIC.

You have to specify the name of the DDM to be deleted. You are then asked to confirm the deletion on a subsequent screen.

To delete multiple DDMs, you specify a DDM name with asterisk notation. This will automatically invoke the function Delete DDMs of the SYSMAIN utility (see DDMs, as described in the section The SYSMAIN Utility in the Natural Utilities for Mainframes documentation).

The contents of the SYSDDM work area is not affected by the deletion.

When you delete a DDM with SYSDDM, the corresponding Natural Security file profile is automatically deleted, too.

List DDMs

This function corresponds to the system command LIST DDM. It displays a list of the DDMs stored in the specified FDIC system file.

From the list, you can select individual DDMs for further processing.

To select a DDM from the list, you mark it with a command in the Cmd column. For information on possible commands, you enter a question mark (?) in the Cmd column.

For information on other options available, see the system command LIST in the Natural Command Reference documentation.

List DDMs with Additional Information

This function displays a list of the DDMs stored in the specified FDIC system file.

From the list, you can select individual DDMs for further processing.

To select a DDM from the list, you mark it with a command in the C column. For information on possible commands, you enter a question mark (?) in the C column.

For each DDM listed, the following information is displayed:

- Database ID, file number, DDM type, DDM length in bytes;
- Security type (only under Natural Security):
Public, Private, Access or Undef(ined);
- File type: Log.View, Phy.File or Log.File for VSAM DDMs;
Userfile for Super Natural DDMs;
- VSAM name;
- Remarks; for example, SupNat (for a Super Natural DDM)
or the VSAM file organization (KSDS, RRDS, ESDS or VRDS).

Copy DDM to Another FDIC File

This function is used to copy DDMs from one system file (FDIC) and/or database to another. This may be necessary, for example, when a Natural application is transferred from test to production status.

This function uses the function for copying DDMs of the SYSMAIN utility (see also DDMs, as described in the section The SYSMAIN Utility in the Natural Utilities for Mainframes documentation).

Show Defined DBIDs and Used FNRs

When you invoke this function, a menu will be displayed from which you can select the following functions:

- Database IDs Defined in Natural
- File Numbers of Existing DDMs for a Database

Database IDs Defined in Natural

This function displays a list of all DBIDs defined in NTDB macros of the Natural parameter module, sorted by database types (the NTDB macro is described in Parameter Modules in the Natural Parameter Reference documentation).

The default database type is shown at the top of the screen. DBIDs of the default database type are **not** listed.

File Numbers of Existing DDMs for a Database

This function displays for a given DBID a list of all file numbers for which DDMs have been defined.

You enter the desired DBID on the menu Show Defined DBIDs and Used FNRs when you invoke the function.

You can also invoke this function by entering a DBID in the command line of the screen Database IDs Defined in Natural and pressing PF5.

SYSMAIN Utility - Overview

The SYSMAIN utility, which is available online and in batch mode, is used to transfer objects within the Natural system from one environment to another.

This documentation covers:

- General Information
- Invoking SYSMAIN
- SYSMAIN Functions and Function Processing
- Direct Commands
- SYSMAIN Parameters and Keywords
- Programming Objects
- Debug Environments
- Error Messages
- Profiles
- Rules
- DDMs
- DLI Subfiles
- Commands Issued to SYSMAIN
- PF Keys
- Data Rejected
- Special Considerations for Natural Administrators

SYSMAIN General Information

This section covers the following topics regarding the SYSMAIN utility:

- Objects
- Environments
- Functions

Objects

The following Natural objects can be transferred with SYSMAIN:

Object	Explanation
Programming Objects	Programs, subprograms, subroutines, classes, maps, data areas (local, parameter and global), copycodes, help routines, expert models, recordings, texts, reports, macros, processors and dialogs.
Debug Environments	User debug environments for online program testing.
Error Message Texts	Short and long texts of Natural system and user-supplied error messages.
Profiles	Editor profiles, map profiles, device profiles, and parameter profiles (created with the SYSPARM utility).
Rules	Automatic and free rules.
DL/I Subfiles	Natural NSBs, NDBs and UDFs.
DDMs	Data definition modules.

Environments

The environment in which a Natural object is located depends on the object type.

The environment for each type of object is defined as follows:

Programming Object	Debug Environment	Error Message	Profile	Rule	DL/I Subfile	DDM
database						
FUSER and FNAT file	FUSER file	FUSER and FNAT file	FNAT file	FDIC file	FDIC file	FDIC file
file name (VSAM only)						
library	library	library				
		language				

Functions

SYSMAIN provides the following functions:

Function	Explanation
COPY	Copy object from one environment to another environment.
DELETE	Delete object from a specific environment.
FIND	Locate a single object within a specific environment.
LIST	Display a range of objects within a specific environment.
MOVE	Transfer object from one environment to another.
RENAME	Give an object a new name, and (optionally) transfer it to a new environment.

Not all functions can be applied to all objects. The following table shows which functions are valid for each type of object:

Function	Programming Object	Debug Environment	Error Message	Profile	Rule	DDM	DL/I Subfile
COPY	X	X	X	X	X	X	X
DELETE	X	X	X	X	X	X	X
FIND	X		X				
LIST	X	X	X	X	X	X	X
MOVE	X	X	X	X	X	X	X
RENAME	X	X	X	X	X		

Invoking SYSMAIN

- Invoking SYSMAIN Online
 - Terminating SYSMAIN Online
 - Invoking SYSMAIN in Batch Mode
 - Invoking SYSMAIN by a Subprogram
-

Invoking SYSMAIN Online

There are two methods for invoking the SYSMAIN utility:

▶ To invoke SYSMAIN online from any Natural library

- Enter the command SYSMAIN.
The current setting of the system variable *LIBRARY-ID is passed to SYSMAIN and used as the default source library for processing of programming objects.
If you issue a Natural system command from SYSMAIN, the command will apply to the library from which SYSMAIN has been invoked.

▶ To invoke SYSMAIN online from the Natural Main Menu

1. Select Maintenance and Transfer Utilities. A corresponding menu is displayed.
2. Select Transfer Objects to Other Libraries.

Terminating SYSMAIN Online

The SYSMAIN utility is terminated with a period (.), with PF3 or with CLEAR from the SYSMAIN main menu; or with a period (.) in the command line of any other SYSMAIN menu.

Do not terminate the SYSMAIN utility with the terminal command %%, because the environment may not be reset correctly.

Invoking SYSMAIN in Batch Mode

The SYSMAIN utility is invoked in batch mode in the same way as it is invoked online; however, one or more direct command strings must follow the SYSMAIN command. Direct commands for each SYSMAIN function as it applies to a specific object are included in sections discussing processing of specific Natural objects. If you want to execute other Natural commands after the SYSMAIN command(s), you must first terminate SYSMAIN by using the commands END or Quit.

There are two ways to enter direct commands:

- The direct command string follows the SYSMAIN command in the same input line; each parameter in the command string can be delimited by a blank character instead of the delimiter.
- The direct command string follows the SYSMAIN command in the next input line; each parameter in the command string must be delimited by the delimiter and **not** by a blank character. If the direct command string is longer than one single line, the CF character (see also the session parameter CF as described in the Natural Parameter Reference documentation) must be placed at the end of the line to continue with the direct command in the next line.

Invoking SYSMAIN by a Subprogram

MAINUSER is a subprogram which allows you to perform the various SYSMAIN functions directly from any user-written object (subroutine, program or subprogram) without going through the normal steps of invoking SYSMAIN. Upon completion of processing of the SYSMAIN functions, the utility is terminated and control is returned to the program, subprogram or subroutine from which the request was issued. MAINUSER can be used in either online or batch mode.

Note:

The maximum DATSIZE used by SYSMAIN during processing is 32 KB, depending on the actual request issued.

MAINUSER must not be located in a user library. You must therefore copy it to library SYSTEM on the files FNAT or FUSER or to any SYS-prefixed library which is the steplib for the application.

MAINUSER is invoked with the CALLNAT statement and its relevant parameters (see the CALLNAT statement in the Natural Statements documentation). The parameters are:

Parameter	Explanation
Command (A250)	The direct command string to be executed by SYSMAIN.
Error (N4)	The return code issued by SYSMAIN at the end of processing to indicate a normal end of processing or an error.
Message (A72)	The message corresponding to the error given online.
Library (A8)	The library containing SYSMAIN. If not specified, the default is SYSMAIN.

MAINUSER is invoked as follows:

```
CALLNAT 'MAINUSER' command error message library
```

An example of a callable routine is the program MAINCALL in the library SYSMAIN.

MAINUSER must **not** be invoked from within the library SYSMAIN.

SYSMAIN Functions and Function Processing

The SYSMAIN functions are executed from menu screens (in menu-driven mode) by entering either an appropriate function code on the menu or direct commands in the command line, or in batch mode with direct commands. Object selection criteria are specified using parameters and keywords.

This section covers the following topics:

- Commands
 - Function Processing
-

Commands

Command	Explanation
COPY	<p>This command copies an object from a source environment to a target environment. The object remains unchanged in the source environment.</p> <p>If the target environment already contains an object with the same name (or in the case of an error message, the same number) as the object to be copied, the specified object is not copied. The REPLACE parameter can be used to overwrite the object in the target environment.</p>
DELETE	<p>This command deletes an object from a source environment.</p> <p>If a class is to be deleted, the system command UNREGISTER is carried out for this class. The class is not deleted if an error occurs in the COM environment.</p> <p>Note: If a DDM is deleted with SYSMAIN, the corresponding Natural Security file profile is also deleted.</p>
FIND	<p>This command locates a specific programming object or error message in a source environment. During online processing, a window showing the library currently being searched is displayed.</p>
LIST	<p>This command displays a range of objects within a range of libraries in a source environment.</p>
MOVE	<p>This command transfers objects from a source environment to a target environment. The object is deleted from the source environment and added to the target environment.</p> <p>If the target environment already contains an object with the same name (or in the case of an error message, the same number) as the object to be moved, the specified object is not moved.</p> <p>If a class is to be transferred, the system command UNREGISTER is carried out for this class. The class is not transferred if an error occurs in the COM environment.</p> <p>The REPLACE parameter can be used to overwrite the object in the target environment.</p>
RENAME	<p>This command gives an object a new name using one of the following two options:</p> <ol style="list-style-type: none"> 1. Rename the object in the source environment. 2. Rename the object and transfer it to another (that is, target) environment. <p>The RENAME function deletes the original object in the source environment; therefore, you are prompted with an option to retain the original object (if the original object is to be retained, it is not modified).</p> <p>If the target environment already contains an object with the same name (or in the case of an error message, the same number) as the object to be renamed, the specified object is not renamed. The REPLACE parameter can be used to overwrite the object in the target environment.</p> <p>Only a single programming object, environment, profile or rule can be renamed using automated processing. If a range of programming objects, environments, profiles or rules is to be renamed, selective processing must be used. A range of error messages can be renamed with automated processing.</p>
HELP	<p>General help information on the SYSMAIN utility is displayed if you enter a question mark (?) in the Code field of any menu.</p> <p>You can obtain field-specific help by placing the cursor in the field in question and pressing PF1 or entering the appropriate help character in the field in question and pressing ENTER.</p>
EXIT	<p>This command terminates the SYSMAIN utility.</p> <p>PF3 and CLEAR also terminate SYSMAIN if they are pressed when the SYSMAIN Main Menu is displayed; however if they are pressed when any other SYSMAIN menu is displayed, the main menu is displayed.</p>

Function Processing

When operating in menu-driven mode, an object and function are selected from the SYSMAIN Main Menu.

```

14:19:40          ***** NATURAL SYSMAIN UTILITY *****          1999-12-01
User SAG              - Main Menu -                               Library SYSMAIN

      Code  Object                                     Code  Function

      A    Programming Objects                         C    Copy
      D    Debug Environments                         D    Delete
      E    Error Message Texts                       F    Find
      P    Profiles                                   L    List
      R    Rules                                       M    Move
      S    DL/I Subfiles                              R    Rename
      V    DDMS                                       ?    Help
      ?    Help                                       .    Exit
      .    Exit

Object Code .. A          Function Code .. _

Command ==>
Enter-PF1---PF2---PF3---PF4---PF5---PF6---PF7---PF8---PF9---PF10--PF11--PF12---
      Help Menu Exit Copy Del Find List Move Ren
    
```

An appropriate subfunction menu is then displayed (see the example below for the copy function for programming objects).

The fields contained on the subfunction menus correspond to SYSMAIN parameters; see SYSMAIN Parameters and Keywords.

```

18:47:52          ***** NATURAL SYSMAIN UTILITY *****          1999-11-18
User SAG          - Copy Programming Objects -          Library SYSMAIN

                Code  Function
                A    Copy All/Individual Objects
                C    Copy only Cataloged Objects
                S    Copy only Saved Objects
                W    Copy only Stowed Objects
                ?    Help
                .    Exit

                Code ..... A          Sel. List ... Y
Object Name ..... *_____          Type ..... _____
                Set Number .. ___          XREF .. N
Source Library ... OLDLIB__          Database .... 10___          File .. 50___
Target Library ... NEWLIB__          Database .... 10___          File .. 60___
Options Replace ... N          Criteria .... N

Command ==>
Enter-PF1---PF2---PF3---PF4---PF5---PF6---PF7---PF8---PF9---PF10--PF11--PF12---
      Help Menu Exit Copy Del Find List Move Ren Fsec Fdic Fnat
    
```

Two other fields contained on the subfunction menus are the command line and Sel. List (selection list).

Command Line

In the command line, you can enter one of the following:

- A direct command for processing a SYSMAIN function.
- A special command to the SYSMAIN utility. This command can be preceded by SET (see Commands Issued to SYSMAIN) or by a reserved command.
- A Natural system command. If the command is not uniquely identifiable as a Natural system command, it should be preceded by two slashes (//) to ensure the correct response from SYSMAIN.

Selection List

The Selection List determines which type of processing is to occur in menu-driven mode:

Y	A Selection List is displayed containing all objects which meet the specified selection criteria. You can select objects to be processed (see Selective Processing). Y is the default.
N	Objects are processed automatically, without display of an intervening Selection List (see Automated Processing).

Once the required object and SYSMAIN function have been selected, you specify the values of the various parameters.

Selective Processing

Selective processing is an online facility which displays a selection list listing all objects meeting the specified selection criteria.

- **Menu-driven mode**
Selective processing is the default type of processing when operating in menu-driven mode (it can be deactivated by entering an **N** in the Sel. List field of the menu).
- **Direct command mode**
To activate selective processing in direct command mode, either include the keyword **HELP** in the *with-clause* of the direct command or enter a question mark (?) as the final character of the object name; see Direct Commands.

You can then select objects from the selection list for processing. The status of each object (for example, Moved, Copied, Renamed, Not Replaced, etc.) is displayed in the Message column after it has been processed.

Specific details for processing each type of object are discussed in subsequent sections of this section.

Online report mode can be used to obtain the SYSMAIN Batch Report online; that is, a list of the objects that were affected by a SYSMAIN function being executed and of the actions performed on each of these objects (see also the appropriate sections on the processing of Natural objects in batch mode later in this section).

If the command BATCH (or BAT) is entered on the command line, SYSMAIN processes the request as if in batch mode. Hence, only the result of each action is present in a report type format. This form of processing also allows you to use the %H terminal command, with which you can obtain a hardcopy of the report (if required).

The functions DELETE and MOVE always delete objects in the source environment. Therefore, during online automated processing, a special confirmation screen is displayed, which gives you the option of continuing or terminating the function.

If the Replace parameter has been set to **Y**, you are given the opportunity to confirm every replace operation **before** it is done. A window is displayed, and you can choose to:

- replace the object indicated,
- not replace the object indicated (default),
- terminate the processing of the function by either entering a period (.) or pressing PF3.

If the target environment already contains an object with the same name as the object to be copied, moved or renamed, the specified object is not processed and processing continues with the next object. The REPLACE parameter can be used to override this feature.

The status of individual objects is not displayed, but an appropriate SYSMAIN message is displayed upon completion of processing. However, if the following message is displayed, it indicates that some objects were not processed:

NAT4893 NORMAL END BUT SOME DATA WERE REJECTED

Error message NAT4810 (see the section Data Rejected) lists reasons why an object may not have been processed. Additional reasons for an object not to be processed are discussed in specific sections relevant to each object type.

Batch mode or selective processing should be used if it is necessary to see the status of each object after it is processed.

Enhanced Selection Criteria

When you select programming objects to be processed by SYSMAIN, in addition to the selection criteria already available, you can also select objects by the date/time, user ID and terminal ID related to their saving or cataloging. For example, you can select only those objects that were cataloged on a specific day between 8:00 and 12:00 by a specific user on a specific terminal, which means that the processing of objects according to the selection criteria is based on all selected criteria as a whole, not on each condition.

When you enter **Y** in the Criteria field in any screen for programming objects, a window is provided where you can enter your additional selection criteria.

```

18:49:47          ***** NATURAL SYSMAIN UTILITY *****          1999-11-18
User SAG          - Copy Programming Objects -          Library SYSMAIN
+-----+
!      --- Additional Criteria ---      !
!                                       !
! Object Type ..... _____ !
! Date/Time From .. _____ !
! Date/Time To ... _____ !
! User ID ..... _____ !
! Terminal ID ..... _____ !
!                                       !
! Command ==>                          !
Code .. !
Object Name .. +-----+
                Set Number .. ___ XREF .. N
Source Library ... OLDLIB__ Database .... 10__ File .. 32__
Target Library ... NEWLIB__ Database .... 10__ File .. 32__
Options Replace ... N      + Criteria .... y

Command ==>
Enter-PF1---PF2---PF3---PF4---PF5---PF6---PF7---PF8---PF9---PF10--PF11--PF12---
      Help Menu Exit Copy Del Find List Move Ren Fsec Fdic Fnat
    
```

Note:

A plus character (+) in front of the Criteria indicates that additional criteria have been specified already.

SYSMAIN Direct Commands

SYSMAIN functions can be executed using direct commands, which can be issued either at the same time the SYSMAIN utility is invoked or from the command line of a menu screen.

Direct commands consist of keywords and parameters (see SYSMAIN Parameters and Keywords).

You can issue a direct command in the following ways:

- **In any library:**
Enter SYSMAIN followed by a direct command string.
- **From the command line of the SYSMAIN menu:**
Enter a direct command string.
- **From a programming object:**
Invoke the subprogram MAINUSER with the direct command string as a parameter.
- **In batch mode:**
Enter SYSMAIN followed by a direct command string.

The below section covers the topics:

- General Direct Command Syntax
- Effects of Direct Commands

General Direct Command Syntax

Below is the general direct command syntax that applies to the SYSMAIN utility. For the syntax and direct commands that applies to each object type, refer to the relevant sections in the SYSMAIN documentation.

```

FUNCTION [OBJECT] object-name [AS new-name]
      { FROM } [LIBRARY] lib-name [where-clause]
      { FM  }
      TO [LIBRARY] lib-name [where-clause] [with-clause]
```

In the function-specific syntax diagrams, either **FROM** or **FM** is shown to make the diagrams easier to read; however, **FM** can always be used as a synonym for **FROM** and vice versa.

where-clause

The *where-clause* is optional. The syntax is:

```

[WHERE] [DBID dbid] [FNR file-nr] [NAME name] [CIPHER cipher ]
[PASSWORD password]
[DICTIONARY (dbid, fnr, psw, ciph)]
[SECURITY (dbid, fnr, psw, ciph)]
[LANGUAGE language]
```

with-clause

The *with-clause* is optional. The syntax is:

```
[WITH] [TYPE type] [FMDATE date] [TODATE date] [FMTIME time]
[TOTIME time] [USER user-id] [TID terminal-id] [XREF xref] [HELP]
[REPLACE] [RCOP] [MON] [EXTEND] [NOPROMPT] [SETUSER user]
[SETNO nr] [VDBID dbnr] [VFNR fnr]
```

Sequence of Syntax Elements

The sequence of the direct command syntax is not completely fixed, thus allowing more flexibility in command entry. The options and rules which apply are:

- *FUNCTION*, *OBJECT* and *object-name* must normally be the first three parameters of the command string (*OBJECT* can be omitted in some cases).
- The *library-name* (for programming objects and error messages) must be specified immediately after the **FROM** and **TO** keywords. (If the optional keyword *LIBRARY* is used, it must be entered between the **FROM** or **TO** keyword and the *library-name*).
- The *where-clause* must always follow the **FROM** or **TO** keyword and the *library-name*; the sequence of the keywords and values within the clause can be specified in any order.
- The keywords and values of the *with-clause* can be specified in any order, and the *with-clause* can be placed in any location within the direct command string, except as in the first three positions.

Effects of Direct Commands

Direct commands can result in:

- the display of a selection list for selective processing of the object if a question mark (?) followed by a blank is entered immediately following the object name, or if the keyword *HELP* is included in the *with-clause*; see Selective Processing;
- the automatic processing of a single object or range of objects; see Automated Processing.

When a direct command is issued, you normally are returned to the library from which the command was issued. For more detailed information on direct commands, see the sections which relate to each type of object.

SYSMAIN Parameters and Keywords

Parameters and keywords are the selection criteria used to identify the objects to be processed by the SYSMAIN utility.

Parameters are applicable online and in batch mode; however when using direct commands, a parameter must normally be preceded, or in some cases replaced, by a keyword.

In the following section, keywords are listed under the parameter to which they correspond. They are also included in the direct command syntax for each command.

Parameters are listed alphabetically in this section. They correspond to input fields on a menu screen. After each parameter, its format and length is indicated in parentheses.

A - P	R - S	T - Z
CRITERIA (A1)	REPLACE (A1)	TARGET CIPHER (A8)
DATE FROM (A10)	RULE NAME (A32)	TARGET DATABASE (N5)
DATE TO (A10)	RULE TYPE (A2)	TARGET FILE (N5)
DDM DBID (N5)	SELECTION LIST (A1)	TARGET LANGUAGE (A9)
DDM FNR (N5)	SET NUMBER (N2)	TARGET LIBRARY (A8)
DDM NAME (A32)	SET USER (A8)	TARGET NAME (A8)
ENVIRONMENT NAME (A8)	SOURCE CIPHER (A8)	TARGET PASSWORD (A8)
ERROR NUMBER FROM (N4)	SOURCE DATABASE (N5)	TERMINAL ID (A8)
ERROR NUMBER TO (N4)	SOURCE FILE (N5)	TIME FROM (A5)
ERROR TYPE (A1)	SOURCE LANGUAGE (A9)	TIME TO (A5)
FDIC (A27)	SOURCE LIBRARY (A8)	USER ID (A8)
FSEC (A27)	SOURCE NAME (A8)	XREF (A1)
NEW NAME (A8)	SOURCE PASSWORD (A8)	
NEW NUMBER FROM (N4)	SUBFILE NAME (A8)	
NEW NUMBER TO (N4)	SUBFILE TYPE (A3)	
OBJECT NAME (A9)		
OBJECT TYPE (A15)		
PROFILE NAME (A8)		
PROFILE TYPE (A3)		

This section also covers the following topics:

- Description of Parameters and Keywords
- Additional Keywords for Direct Commands
- Range Notation for SYSMAIN Parameters

Description of Parameters and Keywords

CRITERIA (A1)

This parameter indicates whether additional selection criteria are to be used. If you are working in menu-driven mode, the following values can be entered in the input field:

?	A window with help information about the Criteria parameter appears.
N	No additional selection criteria are to be used. (This is the default.)
Y	A window is displayed where you can specify the additional selection criteria listed below.

OBJECT TYPE (A15)

Keyword: **TYPE**

The specific type of programming object to be processed (see description of the OBJECT TYPE parameter of the programming objects processing screens).

In menu-driven mode, you can obtain a list of all programming object types available when you enter a question mark (?) in the Type field; a corresponding window appears.

If one or more object types have already been specified on the corresponding programming object processing screen, the OBJECT TYPE parameter of the additional criteria window is already preset with the same specification when the window appears.

DATE FROM (A10)

Keyword: **FMDATE**

Date on which the programming object was cataloged or saved. All programming objects cataloged or saved on or after this date are selected. The date must be specified according to the setting of the DTFORM parameter.

TIME FROM (A5)

Keyword: **FMTIME**

Time on which the programming object was cataloged or saved on a specific date. All programming objects cataloged or saved on or after this date and time are selected. The time must be specified in the format *HH:II* (HH = hours, II = minutes).

DATE TO (A10)

Keyword: **TODATE**

If entered alone (not in conjunction with the Date From parameter), all programming objects cataloged or saved up to this date are selected. The date must be specified according to the setting of the DTFORM parameter.

A date range can be specified by entering Date From and Date To values.

TIME TO (A5)

Keyword: **TOTIME**

If entered alone (not in conjunction with the Time From parameter), all programming objects cataloged or saved up to this time on a specific date are selected. The time must be specified in the format *HH:II* (HH = hours, II = minutes).

A time range can be specified by entering Time From and Time To values.

USER ID (A8)

Keyword: **USER**

All programming objects cataloged or saved by the specified user are selected.

TERMINAL ID (A8)

Keyword: **TID**

All programming objects cataloged or saved on the specified terminal are selected.

DDM DBID (N5)

Keyword: **VDBID**

Specific DBID under which a DDM was cataloged. Only DDMs with this DBID are processed. If this parameter is left blank or a **0** is entered, there is no database verification and all databases are selected.

DDM FNR (N5)

Keyword: **VFNR**

Specific FNR under which a DDM was cataloged. Only DDMs with this FNR are processed. If this parameter is left blank or a **0** is entered, there is no file number verification and all file numbers are selected.

DDM NAME (A32)

The name of the DDM to be processed. See also Range Notation for SYSMAIN Parameters.

ENVIRONMENT NAME (A8)

The name of the debug environment to be processed. See also Range Notation for SYSMAIN Parameters.

ERROR NUMBER FROM (N4)

The number of the error message to be processed. Each Natural or user-supplied error message within a Natural library is uniquely defined by the error number.

ERROR NUMBER TO (N4)

Keyword: **THRU**

Used in conjunction with the Error Number From parameter to specify a range of error numbers.

ERROR TYPE (A1)

Keyword: **TYPE**

The specific type of error message to be processed:

S	short error message
E	extended error message
A	all error message types (default)

Note:

This parameter is only applicable with direct commands. In menu-driven mode, error message types are shown as subfunctions.

FDIC (A27)

Keyword: **DIC**

Specifies the Adabas security for the FDIC source and/or target system file. (See the sections on direct commands for specific objects for details regarding syntax.)

FSEC (A27)

Keyword: **SEC**

Specifies the Adabas security for the FSEC source and/or target system file. (See the sections on direct commands for specific objects for details regarding syntax.)

NEW NAME (A8)

Keyword: **AS**

The name to be given to a programming object, debug environment, profile **or** rule when it is renamed with the RENAME function.

NEW NUMBER FROM (N4)

Keyword: **AS**

The new number to be assigned to an error message when it is renamed with the RENAME function.

NEW NUMBER TO (N4)

Keyword: **THRU**

Used in conjunction with the New Number From parameter to specify a range of error numbers.

If you are renumbering a range of error messages within the same environment, the range values must not overlap. For example, it is not possible to rename error numbers 1 - 6 as new error numbers 5 - 10.

OBJECT NAME (A9)

The name of the programming object to be processed. If the LIST function is selected, this parameter is referred to as **OBJECT NAME START VALUE** (see also Range Notation for SYSMAIN Parameters).

OBJECT TYPE (A15)

Keyword: **TYPE**

The specific type of programming object to be processed:

P	program
N	subprogram
S	subroutine
M	map
H	helproutine
Y	expert model
R	report
A	parameter data area
G	global data area
L	local data area
C	copycode
T	text
Z	recording
O	ISPF macro
3	dialog
4	class
5	processor
*	all programming object types (default)

Note:

You can specify several types at the same time and in any sequence. For example, if you specify PAM, programs, parameter data areas and maps are processed.

If one or more object types have already been specified in the window for additional selection criteria, the same specification is displayed for the OBJECT TYPE parameter in the corresponding programming object processing screen, once you have left the window (see also the OBJECT TYPE parameter as part of the enhanced selection criteria).

PROFILE NAME (A8)

The name of the profile to be processed. See also Range Notation for SYSMAIN Parameters.

PROFILE TYPE (A3)

Keyword: **TYPE**

The specific type of profile to be processed:

E	editor profile
D	device profile
M	map profile
P	parameter profile
*	editor, device and map profiles (default)

REPLACE (A1)

Keyword: **REP**

Replace option for an object which is being moved, copied or renamed. If you are working in menu-driven mode, the following values can be entered in the input field:

Y	An object with the same name which is already present in the Target library is to be replaced.
N	An object with the same name which is already present in the Target library is not to be replaced. N is the default.

Note:

If a programming object is replaced it is also deleted from the Natural buffer pool; any existing cross-reference records are also deleted if Predict is installed.

RULE NAME (A32)

The name of the rule (automatic or free rule) to be processed. See also Range Notation for SYSMAIN Parameters.

RULE TYPE (A2)Keyword: **TYPE**

The specific type of rule to be processed:

A	automatic rule
F	free rule
AF	both automatic and free rules

SELECTION LIST (A1)Keyword: **HELP**

Indicates whether automated or selective processing is to apply to an object. If you are working in menu-driven mode, you can enter the following values in the input field:

Y	A Selection List is displayed containing all objects which meet the specified selection criteria. You can select objects to be processed; see Selective Processing. Y is the default.
N	Objects are processed automatically, without display of an intervening selection list; see also Automated Processing.

SET NUMBER (N2)Keyword: **SETN**

Supports Predict Sets. Number of the retained Set created with the Predict XREF Save Set facility. You can apply all SYSMAIN processing functions to the objects included in this Set.

If any valid number is specified, SYSMAIN assumes a Predict Set. If no number is specified, normal object processing is assumed.

SET USER (A8)Keyword: **SETU**

Provides the possibility to overwrite the User ID specification for a Predict Set as a part of the security for Predict files. If you are working in menu-driven mode, the security for Predict (FDIC) Files screen is invoked by entering the SET FDIC command or by pressing PF11 (Fdic) in any programming object processing screen.

Note:

SET USER is only evaluated if a valid number has been specified for SET NUMBER.

SOURCE CIPHER (A8)

Keyword: **CIPH**

Cipher key of the Source file (used in the *where-clause*).

SOURCE DATABASE (N5)

Keyword: **DBID**

The number of the database (1 - 253) which contains the object to be processed.

SOURCE FILE (N5)

Keyword: **FNR**

The number (1 - 255) of the Natural FNAT, FDIC or FUSER file which contains the object to be processed.

SOURCE LANGUAGE (A9)

Keyword: **LANG**

The language in which the error message is written.

Each error message within each library can exist in 1 to 60 languages. The languages can be specified using any combination of language codes.

For information on which language code is assigned to which language, see the Natural system variable *LANGUAGE in the Natural Programming Reference documentation.

To select error messages in all existing languages, asterisk notation (*) can be used.

SOURCE LIBRARY (A8)

Keyword: **LIB**

The name of the library which contains the object to be processed or to which the error message is assigned.

If error messages are processed and the field is left blank, the Natural system error messages are processed.

SOURCE NAME (A8)

Keyword: **NAME**

The DDNAME/FCT entry for the Source file number (VSAM only).

SOURCE PASSWORD (A8)

Keyword: **PSW**

Password for the Source file (used in the *where-clause*).

SUBFILE NAME (A8)

The name of the DL/I subfile to be processed. See also Range Notation for SYSMAIN Parameters.

SUBFILE TYPE (A3)

Keyword: **TYPE**

The specific type of DL/I subfile (Natural NSB, NDB or UDF) to be processed.

TARGET CIPHER (A8)

Keyword: **CIPH**

Cipher key of the Target file (used in *where-clause*).

TARGET DATABASE (N5)

Keyword: **DBID**

The number of the database into which the object is to be moved or copied.

TARGET FILE (N5)

Keyword: **FNR**

The number of the Natural system file or Predict file into which the object is to be moved or copied.

TARGET LANGUAGE (A9)

Keyword: **TO, LANG**

The language in which the error message is to be written.

Each error message within each library can exist in 1 to 60 languages. The languages can be specified using any combination of language codes. For information about language codes, see the system variable *LANGUAGE in the Natural Programming Reference documentation.

TARGET LIBRARY (A8)

Keyword: **LIB**

The name of the library into which the object is to be placed or to which the error message is to be assigned.

If error messages are processed and the field is left blank, the Natural system error messages are processed.

TARGET NAME (A8)

Keyword: **NAME**

The DDNAME/FCT entry for the Target file number (VSAM only).

TARGET PASSWORD (A8)

Keyword: **PSW**

Password for the Target file (used in the *where-clause*).

XREF (A1)

Keyword: **XREF**

Indicates whether SYSMAIN is to support XREF data stored on Predict system files.

N	Cross-reference data are not processed, except when using the DELETE function. If a cataloged object is deleted, SYSMAIN always deletes any existing XREF data for this object.
Y	All cross-reference data are processed.
S	A specified object is processed regardless of whether it has cross-reference data or not. Any existing XREF data are processed, which means that the fact that objects have or have no XREF data is no criterion for rejecting the processing of an object.
F	All cross-reference data are processed and the object must be documented in Predict.

Additional Keywords for Direct Commands

In addition to the keywords shown with the parameters above, the following keywords can also be used with direct commands to specify selection criteria:

Keywords	Explanation
ALL	All saved and/or cataloged programming objects are selected for processing.
CAT	All cataloged programming objects are selected for processing. (Any corresponding saved programming object is not processed.)
<u>EXTEND</u>	Refers to the List Objects function. If EXTEND is specified, the saved object is also displayed; if EXTEND is not specified, only the object name and directory information are displayed (in batch mode only).
HELP	Activates online selective processing.
IN/FM	Refers to a source environment.
MON	Activates online trace facility.
<u>NOPROMPT</u>	Suppresses all prompts.
<u>RCOP</u>	Used with direct commands to specify that a copy of the object being renamed is to be made.
<u>SAVED</u>	All saved programming objects are selected for processing. (Any corresponding cataloged object is not processed.)
STOWED	All programming objects which are both saved and cataloged are selected for processing.
TO	Refers to a target environment.
WITH	Optional keyword to indicate the start of a <i>with-clause</i> .
WHERE	Optional keyword to indicate the start of a <i>where-clause</i> .
.	End of command. If this character is detected anywhere within a command string, all subsequent data are ignored.

Range Notation for SYSMAIN Parameters

All SYSMAIN functions allow the Environment Name, Object (Source) Name, Profile Name, Rule Name, Subfile Name and DDM Name parameters to be specified as a range.

In addition, the "Object Start Value" and "Reposition to" fields for the FIND and LIST functions also specify a range.

With the FIND and LIST functions, also a range of libraries with programming objects can be specified with the Source Library parameter. The same applies to the LIST function with debug environments and to the FIND function with error messages. Using this option, however, can be rather time-consuming depending on how often the selection criteria occur.

You can use the following range notation:

<i>value*</i>	<p>All libraries or objects whose names begin with <i>value</i> are processed.</p> <p>Example: A value of MENU* results in processing of all libraries or objects with a name beginning with MENU, such as MENU1, MENUOFF and MENUX.</p> <p>Note: A question mark (?) has the same effect as an asterisk (*).</p>
<i>value></i>	<p>All libraries or objects whose names begin with a value greater than or equal to <i>value</i> are processed.</p> <p>Example: A value of MEN> results in processing of all libraries or objects whose name begins with a value greater than or equal to MEN, such as MENU, ORDER, SYSDDM and TRS.</p>
<i>value<</i>	<p>All libraries or objects whose names begin with a value less than or equal to <i>value</i> are processed.</p> <p>Example: A value of MEN< results in processing of all libraries or objects whose name begins with a value less than or equal to MEN, such as MAINMENU, CMD and ADALOG.</p>

SYSMAIN Programming Objects

All SYSMAIN functions can be performed on programming objects.

In environments in which different FNAT and FUSER files are used, programming objects are stored in the files according to the name of the library in which they are contained. If the library begins with SYS (except for the library SYSTEM), objects are stored in the FNAT files. In all other libraries, the objects are stored in the FUSER files.

The section below covers the following topics:

- Programming Object Menus
- Processing Status
- Direct Commands for Programming Objects
- Programming Objects in Batch Mode
- XREF Considerations for Programming Objects

Programming Object Menus

The subfunction menu displayed depends on the function selected.

The COPY subfunction menu is shown as the example in this documentation; however, menus for other functions are similar in format.

```

17:54:34          ***** NATURAL SYSMAIN UTILITY *****          1999-11-25
User SAG          - Copy Programming Objects -          Library SYSMAIN
                                                Recat: ON

                Code  Function
                A  Copy All/Individual Objects
                C  Copy only Cataloged Objects
                S  Copy only Saved Objects
                W  Copy only Stowed Objects
                ?  Help
                .  Exit

                Code ..... A          Sel. List ... Y
Object Name ..... *_____          Type ..... _____
                Set Number .. ___          XREF .. N
Source Library ... OLDLIB__          Database .... 10___          File .. 32___
Target Library ... NEWLIB__          Database .... 10___          File .. 51___
Options Replace ... N          Criteria .... N

Command ==>
Enter-PF1---PF2---PF3---PF4---PF5---PF6---PF7---PF8---PF9---PF10--PF11--PF12---
      Help Menu Exit Copy Del Find List Move Ren Fsec Fdic Fnat
    
```

Note:

If the profile parameter RECAT=ON has been set, this is indicated on the menus.

The functions listed on the menus for programming objects have the following meanings:

Function	Explanation
A	Process All/Individual Objects Process any object which exists as a saved, cataloged or stowed object.
C	Process only Cataloged Objects Process any object which exists as a cataloged object.
S	Process only Saved Objects Process any object which exists as a saved object.
W	Process only Stowed Objects Process only objects which exist in both saved and cataloged form. The exception to this is copycode and recording, neither of which can be cataloged. They are, however, included in processing when this option is specified.

If the profile parameter RECAT has been set to ON for the Natural session, normal dynamic recatalog option rules apply (see the RECAT profile parameter in the Natural Parameter Reference documentation). The following considerations apply:

- If an object exists in stowed format, neither the saved nor the cataloged object can be processed independently of one another.
- If an object exists only in cataloged format, it may not be possible to select it on the Selection List screen and thus it cannot be processed.
- When automated processing is being used, any object not satisfying these rules is immediately ignored and processing continues with the next object.

If selective processing has been requested, a selection list is displayed. If there are more programming objects in a library than can be displayed on a single screen, the additional objects can be displayed by pressing ENTER.

Select specific programming objects for processing by entering the desired option in column **C** (code) to the left of the object names. A help character (?) in this column invokes a window listing the options available for the specific function. These options make it possible to perform additional functions before actively processing the object.

The options which can be entered on a COPY, DELETE, FIND, LIST, MOVE or RENAME selection list have the following meanings.

Option	Explanation
A	Process Cat. and/or Sav. Object Process all object types listed in the corresponding S/C column; that is, saved (S) objects, cataloged (C) objects or stowed (S/C) objects.
C	Process Cataloged Object only Process only the cataloged form of the object, even if there is a corresponding saved object. If C is specified for an object which exists only as a saved object, an error occurs.
S	Process Saved Object only Process only the saved form of the object, even if there is a corresponding cataloged object. If S is specified for an object which exists only as a cataloged object, an error occurs.
B	Delete Object in Buffer Pool Select objects and optionally delete them from the Natural buffer pool. Deletion of the specified object(s) must be confirmed by entering DELETE in a window that appears once you have specified the object(s) and pressed ENTER.
H	Hardcopy of Saved Object Make a hardcopy of the saved object.
I	List Directory Information Review a cataloged and/or saved object before processing it. It lists directory information and corresponds to the system command LIST DIR.
L	List Saved Object Review the source code of an object before processing it (valid only with saved objects).
X	Export Object to PC Download Natural source objects to a personal computer (see also below) Note: This option is applicable only if Natural Connection is installed. To be able to use this option, the work files 6 and 7 must be specified as PC work files.
Z	Calculate Sizes Review the various sizes of the saved and cataloged objects, for example, the DATSIZE, ESIZE and MCG size.

If you have Natural Connection installed and choose to download Natural source objects to a PC (that is, to specify option X), a window will be displayed in which you specify the following:

Option	Explanation
Drive	The name of the PC drive to which you want the object(s) to be downloaded.
Path	The path name of the PC directory to which you want the object(s) to be downloaded.
Extension	The name of the extension of the specified PC path. If you specify NS*, the asterisk (*) will be replaced by the specified object type.

Note:

If the specified directory does not exist, you will receive an appropriate error message.

With the FIND and LIST functions, options **A**, **C** and **S** are not available. However, it is possible to list libraries. The Library Selection screen is displayed either when you leave the Find Selection or the List Selection screen, or when you specify a range of libraries in the FIND or LIST processing screens.

On the Library Selection screen, you can enter the following options:

Option	Explanation
D	<p>Display Objects in a Lib. (Short)</p> <p>Short list of programming objects in the specified library. The listing of an object only shows the name, the saved/cataloged specification and the type of the object.</p>
L	<p>List Objects in a Lib. (Extended)</p> <p>Extended list of programming objects in the specified library. The listing shows one object per line with the corresponding directory information.</p>
R	<p>Verify Subroutine Usage</p> <p>Provide a list at object level that informs you of the external subroutines used by each object in the specified library. All external subroutines performed by any object are listed, as well as the names of the corresponding cataloged subroutines. If an object is a cataloged subroutine itself, the name of the external subroutine it contains is also provided.</p>
S	<p>List all Subroutines in a Library</p> <p>List the names of all external subroutines used in a library.</p>

If the FIND or LIST function is used, the current setting of the field Object Start Value is of importance. The Object Start Value can be set to blank or an asterisk (*) to list the entire library; or a range notation can be used. If the library being listed contains no objects which satisfy the range criteria specified, an appropriate message is displayed. This option is useful with large libraries in order to restrict the number of objects being displayed.

The Object Start Value is valid in online mode only, and only when an actual library is to be listed. It is not a selection criterion for the LIST function.

Processing Status

After the individual objects have been selected on the selection list, they are processed by SYSMAIN and an appropriate message is displayed in the message column.

Options H, I and L

If the **H**, **I** or **L** option was selected, the message returned upon completion of processing is Printed, Directory or Listed.

Options B, X and Z

If the **B** option was selected, the message returned upon completion of processing is either "NBP deleted" or Ignored, depending on whether deletion from the buffer pool for the specified object(s) has been confirmed or not.

If the **X** option was selected, the message returned upon completion of processing is Exported, and if the **Z** option was selected, the message is Sized.

Options A, C and S

If the **A**, **C** or **S** option was selected, the message returned upon completion of normal processing is Copied, Deleted, Moved or Renamed, depending on the function selected.

An object can be processed again using the **A**, **C**, or **S** options, after the **B**, **H**, **I**, **L**, **X** or **Z** option processing has been completed.

Other messages which may be returned are:

Message	Explanation
Replaced	The Replace option was set to Y and the Target object was deleted before the COPY, MOVE or RENAME function was completed.
Not Replaced	The Replace option was set to N and an object with the same Object Name already exists in the target environment. The function was not completed.
Subrtn Exists	The cataloged subroutine had an external subroutine name already used by another cataloged subroutine in the target environment. The function was not completed.
Class Exists	The cataloged class had an external class name already used by another cataloged class in the target environment. The function was not completed.
GUID Exists	The cataloged class had a GUID already used by another cataloged class in the target environment. The function was not completed.
LRec-Err: <i>nn</i>	An error was returned for the class during copying of the class link records. The function was not completed. Possible values: 3 = inconsistency in class records, 4 = no record found.
DB Error: <i>nnn</i>	A database error was returned for the object during processing. The function was not completed.
In Use	An Adabas response code 145 was returned during the UPDATE/ READ processing of an object. The function was not completed.
No Xref	A cataloged object was being processed and the XREF parameter was set to Y or F . No XREF data exist on the FDIC file specified for the object. The function was not completed.
Err NAT2999	A cataloged object was being processed with the XREF parameter set to F . No Predict entry exists on the FDIC file specified for the object. The function was not completed.
Exit: <i>nnn</i>	A user exit routine was active and a non-zero return code was returned by the exit (<i>nnn</i> = the return code). The function was not completed. See also under User Exits.
< = Found	Marks the objects found in one or more libraries by a FIND function (when the library is listed).
Invalid Date	The FROM or TO dates of the saved and/or cataloged object do not conform to the normal format. The object was not processed. Resave or catalog the object to validate the dates.

Direct Commands for Programming Objects

The direct command syntax for processing of programming objects is shown in this section. (Since the *where-clause* and *with-clause* syntax are identical for each command, they are only shown once with the COPY and MOVE command syntax below.)

COPY and MOVE Direct Command Syntax

[COPY MOVE]	}	ALL CATALOGED SAVED STOWED	}	<i>name</i> FM [LIBRARY] <i>lib-name</i> [<i>where-clause</i>] TO [LIBRARY] <i>lib-name</i> [<i>where-clause</i>] [<i>with-clause</i>]
------------------------	---	---	---	---

where-clause

[WHERE] [DBID <i>dbid</i>] [FNR <i>file-nr</i>] [NAME <i>name</i>] [CIPHER <i>cipher</i>] [PASSWORD PSW] <i>password</i> [DIC (<i>dbid, fnr, psw, ciph</i>)] [SEC (<i>dbid, fnr, psw, ciph</i>)
--

Note:

Commas must be used as separators between the values following the DIC and SEC keywords; or if a value is missing. For example: DIC(10,,secret,2a). If the Natural session parameter ID has been set to a comma, use a slash (/) sign as the separator between values.

If no DBID or FNR is specified, and SYSMAIN is called via the system command SYSMAIN or via the subprogram MAINUSER, the following rules apply:

The DBID and FNR of the file from which SYSMAIN was called are always used. If, for example, you enter the command SYSMAIN in the FUSER library, the DBID and FNR of the file are used.

with-clause

```
[WITH] [TYPE type] [FMDATE date] [TODATE date] [FMTIME time]
[TOTIME time] [USER user-id] [TID terminal-id] [XREF xref]
[REPLACE] [RCOP] [HELP] [MON] [NOPROMPT]
[SETUSER user-id SETNO set-number] [EXTEND]
```

Examples:

```
COPY PROG1 FM TESTORD TO ORDERS DBID 1 FNR 6 REP
C PGM* WITH REP TYPE PNS FM PRODLIB TO TESTLIB
```

```
M PROG1 TO NEWLIB
MOVE STOWED * TO NEWLIB WHERE DBID 100 FNR 160 FMDATE 02-11-01 FM OLDLIB
WITH XREF Y
```

DELETE Direct Command Syntax

```
DELETE { ALL
        CATALOGED
        SAVED
        STOWED } name [IN [LIBRARY] lib-name [where-clause]] [with-clause]
```

Examples:

```
DELETE CAT M> IN LIB ORDERS
DEL * IN TESTLIB DBID 1 FNR 5 NAME SYSNAT
D SA * LIBTEST TYPE GLA
DEL * TYPE PM IN TESTORD FMDATE 02-01-01 TODATE 02-05-31
```

FIND, LIST and LISTLIB Direct Command Syntax

<pre> { FIND LIST LISTLIB } </pre>	<pre> { ALL CATALOGED SAVED STOWED } </pre>	<pre> name IN [LIBRARY] lib-name [where-clause] [with-clause] </pre>
--	---	--

Note:

The direct command LISTLIB is only available in batch mode and is used to obtain a list of library names.

Examples:

```

FIND SAVED MENU
FIND STOWED MAINMENU IN SYS* WHERE DBID 1 FNR 5
FIND ALL PROG2 IN PROD* FNR 27 DBID 1

```

```

LIST * IN lib-name
LIST DT* IN lib-name
L SAVED TEST* IN lib-name TYPE PNS FNR 6
L SA TEST* TYPE PM IN lib-name FNR 6 DBID 2 FMDATE 02-01-01

```

```

LISTLIB ALL MENU IN SYS* DBID 10 FNR 44

```

RENAME Direct Command Syntax

<pre> RENAME </pre>	<pre> { ALL CATALOGED SAVED STOWED } </pre>	<pre> name AS new-name [with-clause] FM [LIBRARY] lib-name [where-clause] TO [LIBRARY] lib-name [where-clause] </pre>
---------------------	---	---

Examples:

```

RENAME PGM1 AS PROG1
REN PGM1 AS PROG1 FM TESTLIB DBID 1 FNR 5 TO PRODLIB DBID 2 FNR 6
REN PGM* TYPE PS RCOP FM TESTLIB TO PRODLIB

```

Programming Objects in Batch Mode

In batch mode, direct commands have to be used to process programming objects.

A report is provided to show the status of programming objects processed in batch mode.

Note:

By using Online Report Mode, you can obtain the SYSMAIN batch report online. If required, you can also obtain a hardcopy of the report using the %H option.

XREF Considerations for Programming Objects

All XREF data stored in the Predict system file can also be processed with SYSMAIN. The XREF parameter on the COPY, MOVE and RENAME menus indicates whether SYSMAIN processes XREF data. XREF data are always deleted if the DELETE function is performed.

If Predict Version 2.3 has not been installed, set the XREF indicator to **N** and thus no validation of Predict files is performed. If the FDIC file(s) being used are not valid Predict files, an error message is returned.

The rules for setting the XREF indicator are the same as the ones imposed by Natural Security, however in a non-security environment there are no restrictions.

XREF set to N

If the XREF indicator is set to **N** (no), no XREF data are processed, but in situations where a programming object is deleted, SYSMAIN deletes the XREF data. The target Predict system file is determined according to the current settings of the source or target definition in SYSMAIN FDIC. The default is the value assigned to the Natural parameter FDIC at the start of the Natural session.

XREF set to Y or F

If the XREF indicator is set to **Y** (yes) or **F** (force), the following actions are applied during processing:

- SYSMAIN verifies that XREF data already exist in the Predict system source file.
- If the replace option is active (REPLACE=Y) and a programming object is to be deleted from the target environment, XREF data are deleted from the Predict system target file.
- If a programming object is being copied to a new environment, the XREF data of the programming object are copied from the Predict system source file to the Predict system target file. The library name is changed accordingly and in the case of the RENAME function, the object name is also changed.
- If the MOVE function was requested, the XREF data of the programming object are deleted from the Predict system source file.

XREF set to F

If the XREF indicator is set to **F** (force), SYSMAIN additionally checks that the programming object (program, subroutine, subprogram, map or help routine only) has a Predict program entry defined on the Predict system target file. If not, processing of the object is terminated.

XREF set to S

If the XREF indicator is set to **S** (special), the special case applies where a range of specified objects is processed with corresponding XREF data regardless of whether all of the objects have cross-reference data or not: the objects that have cross-reference data are processed with their cross-reference data, and the objects that have none are also processed.

Errors

If any of the following inconsistencies occur during the SYSMAIN processing of XREF data, all processing for the object or function is terminated and an error message is displayed:

- the value of the XREF indicator in Natural Security is **F** or **Y** and you specified a value of **Y** or **N** respectively,
- the XREF indicator is set to **F** and SYSMAIN finds no documented program entry in Predict for the object being processed,
- an invalid Predict file is specified,
- the value of the XREF indicator in Natural Security is **F** or **Y** and you specified a value of **S**.

SYSMAIN Debug Environments

All SYSMAIN functions, except the FIND function, can be performed on debug environments.

The debug environment specification must always correspond to the DBID and FNR of the relevant FUSER file.

The section below covers the following topics:

- Debug Environment Menus
- Processing Status
- Direct Commands for Debug Environments
- Debug Environments in Batch Mode

Debug Environment Menus

The Debug Environments menu contains all of SYSMAIN functions for the processing of debug environments (there are no subfunction menus for debug environments):

```

18:40:20          ***** NATURAL SYSMAIN UTILITY *****          1999-11-18
User SAG              - Debug Environments -                      Library SYSMAIN

          Code  Function

          C  Copy   Debug Environments
          D  Delete Debug Environments
          L  List   Debug Environments
          M  Move   Debug Environments
          R  Rename Debug Environments
          ?  Help
          .  Exit

          Code ..... C          Sel. List ... Y
Environment Name .. TEST1_____ New Name .... _____
Source Library ... OLDLIB__      Database .... 10__ File .. 32__
Target Library ... NEWLIB__      Database .... 10__ File .. 32__
Options Replace ... N

Command ==>
Enter-PF1---PF2---PF3---PF4---PF5---PF6---PF7---PF8---PF9---PF10--PF11--PF12---
          Help Menu Exit Copy Del          List Move Ren  Fsec          Fnat
    
```

If selective processing has been selected, a selection list is displayed. If there are more debug environments in a library than can be displayed on a single screen, the additional debug environments can be displayed by pressing ENTER.

Select specific debug environments for processing by entering the desired options in column C (code) to the left of the debug environment name. A question mark (?) in this column invokes a window listing the options available for the specific function. With these options you can override the current subfunction setting and perform additional functions before actively processing the environment.

Note:

When a debug environment is moved or copied from one library to another, it must be reinstated in the target library. This can simply be done by modifying the corresponding break point or watchpoint.

Processing Status

After the individual debug environments have been selected on the selection list with option **A** (Process Debug Environment), they are processed by SYSMAIN and an appropriate message (that is, Copied, Deleted, Moved or Renamed) is displayed in the message column depending on the function selected.

Other messages which may be returned are:

Message	Meaning
Exit: <i>nnn</i>	A user exit routine was active and a non-zero return code was returned by the exit (<i>nnn</i> = the return code); function not completed.
Replaced	Replace option was set to Y and the target debug environment was deleted before the COPY, MOVE or RENAME function was completed.
Not Replaced	Replace option was set to N and a debug environment with the same debug environment NAME already exists in the target environment; function not completed.
Invalid	An invalid code was specified for one of the debug environments listed.
Name Error	Rename function was used, but the new name specified was found to be invalid. Either no code was specified for the selection or the specified name either contained invalid special characters or did not start with an alphabetic character.

Direct Commands for Debug Environments

The direct command syntax for processing of debug environments is shown in this section. (Since the *where-clause* and *with-clause* syntax are identical for each command, they are only shown once with the COPY and MOVE command syntax below.)

COPY and MOVE Direct Command Syntax

```
[COPY  
MOVE] DEBUG name FM [LIBRARY] lib-name [where-clause]  
                TO [LIBRARY] lib-name [where-clause] [with-clause]
```

where-clause

```
[WHERE] [DBID dbid] [FNR file-nr] [NAME name]  
[CIPHER cipher] [PASSWORD password]  
                [PSW]
```

with-clause

```
[WITH] [REPLACE] [HELP] [RCOP] [MON]
```

Examples:

```
COPY D env1 FM oldlib WHERE DBID 1 FNR 5 TO newlib WHERE DBID 2 FNR 5 WITH REP  
COPY DEBUG env FM lib1 FNR 6 TO lib2 FNR 7 REP  
MOVE DEBUG env2 FM lib1 WHERE DBID 1 FNR 5 TO lib2 WHERE DBID 2 FNR 5  
MOVE DEBUG env1 FM oldlib FNR 6 TO newlib FNR 7 REP
```

DELETE Direct Command Syntax

```
DELETE DEBUG name [IN [LIBRARY] lib-name [where-clause]] [with-clause]
```

Examples:

```
DEL DEBUG U* IN lib-name FNR 150  
DEL DEBUG TEST* IN lib-name IN DBID 177 FNR 205
```

LIST Direct Command Syntax

```
LIST DEBUG name [IN [LIBRARY] lib-name [where-clause]] [with-clause]
```

Examples:

```
LIST DEBUG env* IN lib-name DBID 1 FNR 5
LIST DEBUG DT* IN lib-name DBID 10
```

RENAME Direct Command Syntax

```
RENAME DEBUG name AS new-name [with-clause]
IN [LIBRARY] lib-name [where-clause]
TO [LIBRARY] lib-name [where-clause]
```

Examples:

```
R DE env1 AS env2 RCOP
REN DEBUG env1 AS env2 IN lib-name DBID 1 FNR 4 TO lib-name DBID 1 FNR 5
REN DEBUG env1 AS newenv IN lib-name FNR 4 TO lib-name FNR 5 REPLACE RCOP
```

Debug Environments in Batch Mode

Direct commands must be used to process debug environments in batch mode.

A report is provided to show the status of debug environments processed in batch mode.

Note:

By using Online Report Mode, you can obtain the SYSMAIN batch report online. If required, you can also obtain a hardcopy of the report by using the %H option.

SYSMAIN Error Messages

All SYSMAIN functions can be performed on error messages. Only authorized users can modify Natural system error messages if Natural Security is installed.

In environments in which different FNAT and FUSER files are used, error messages are stored in the files according to the type of error: Natural system error messages are stored in the FNAT file; user error messages are stored in the FUSER file of the corresponding library.

This section covers the following topics:

- Error Message Menus
- Language Parameter Considerations
- Renumber an Error Message
- Processing Status
- Direct Commands for Error Messages
- Error Messages in Batch Mode

Error Message Menus

The subfunction menu displayed depends on the function selected. The COPY subfunction menu is shown as the example in this documentation; however, menus for other functions are similar in format.

```

18:43:21          ***** NATURAL SYSMAIN UTILITY *****          1999-11-18
User SAG          - Copy Error Message Texts -          Library SYSMAIN

                Code  Function

                A   Copy Short and/or Extended Texts
                E   Copy only Extended Texts
                S   Copy only Short Texts
                ?   Help
                .   Exit

                Code ..... A          Selection List ..... Y

Error   No. From .. ____   No. To .. ____
Source  Library ... _____   Lang. ... *_____   DBID .. 10___   FNR .. 50___
Target  Library ... _____   Lang. ... *_____   DBID .. 10___   FNR .. 60___
Options Replace ... N

Command ==>
Enter-PF1---PF2---PF3---PF4---PF5---PF6---PF7---PF8---PF9---PF10--PF11--PF12---
      Help Menu Exit Copy Del Find List Move Ren Fsec Fn timer
    
```

Note:

For the processing of Natural system error messages, a library name must not be specified. If the source or target library name contains a question mark (?), an asterisk (*), a greater than (>) or less than (<) sign, a selection list is displayed with libraries containing error messages (except if the FIND function was used).

The functions listed on the menus for error messages have the following meanings:

Function	Explanation
A	Process Short and/or Extended Texts Process any error message which exists in short and/or extended form.
E	Process only Extended Texts Process any error message which exists in extended form. Only extended error messages with corresponding short error messages are processed. Note: An extended error message cannot be transferred to a target environment if there is no corresponding short error message in the target environment.
S	Process only Short Texts Process any error message which exists in short form.

If selective processing has been selected, a selection list is displayed. If there are more error messages in a library than can be displayed on a single screen, the additional messages can be displayed by pressing ENTER.

Select specific error messages for processing by entering the desired option in column C (code) to the left of the error number. A question mark (?) in this column invokes a window listing the options available for the specific function. These options make it possible to perform additional functions before actively processing the object.

The options which can be entered on an error message selection list have the following meanings:

Option	Explanation
A	Process Extended and Short Text Process all error message types listed in the corresponding Type column, which means, short (S), extended (E) or short and extended (S/E) messages.
E	Process only Extended Error Text Process only the extended form of the error message. If there is no corresponding short error message in the target environment, the extended error message cannot be processed completely. If E is specified for a message which exists only as a short message, an error is returned.
S	Process only Short Error Text Process only the short form of the error message.
L	List Extended and Short Text Review an error message before processing it. The short and/or extended text is displayed, depending on the subfunction previously specified. The default language corresponds to the codes or values in the Source Language parameter or, if no match is found, to the setting of the system variable *LANGUAGE.

Language Parameter Considerations

Attention must be given to the setting of the Source and Target Language parameters. These have three possible "logical" settings:

1. If an asterisk (*) is specified, all languages are treated as one unit. All languages are processed.
For example, if the source error message exists only in languages 1, 2 and 3, and for the target error message only languages 1, 4 and 6 are defined, after a COPY function, the resulting target message exists only in languages 1, 2 and 3.
2. If the language parameters are specified as individual codes, each occurrence of language in the parameter is processed individually.
For example, if the source error message contains languages 1, 2 and 3, and the parameter is set to123, and if the target error message contains languages 1, 4 and 6, and the parameter is also set to123, after a COPY function, the resulting target error message contains languages 1, 2, 3, 4 and 6, but only the English target error message is overwritten by the English text of the source error message.
3. If a single language code is specified for the source error message, and multiple language codes are specified for the target error message, the resulting target error message is in the language specified for the source upon completion of the COPY function.
For example if the source error message contains language 1 and the target error message contains languages 23456, the resulting error message is in language 1 for all the specified languages. This allows the use of the *LANGUAGE parameter.

Renumber an Error Message

Error messages can be moved from one library to another, or the languages for each error message can be copied, moved or replaced with the MOVE and COPY functions. In addition, it is possible to renumber a single error message or renumber a range of error messages. This can be done with the RENAME function.

When specifying the ranges, the number of error messages specified by the range "From - To" range of the Source Library must be equal in number to the "From - To" range of the Target Library. In addition, if renumbering error messages within a single library, there must be no overlapping in the New Number range.

Examples:

The following examples are valid:

REN ERROR 1 THRU 100 AS 101 THRU 200 IN CLAIMS

REN ERROR 101 THRU 200 AS 1 THRU 100 IN CLAIMS

The following examples are invalid, because the number ranges are overlapping:

REN ERROR 1 THRU 100 AS 51 THRU 150 IN CLAIMS

REN ERROR 101 THRU 200 AS 51 THRU 150 IN CLAIMS

When large ranges of error messages are being processed, the processing of error messages may require significant resources. In such cases, batch-mode processing may be preferable.

Processing Status

After the individual error messages have been selected on the selection list, they are processed by SYSMAIN and an appropriate message is displayed in the message column.

Option L

If the **L** option was selected, the message returned upon completion of processing is Listed.

An object can be processed again using the **A**, **E** or **S** options after the **L** option processing has been completed.

Options A, E and S

If the **A**, **E** or **S** options were selected, the messages returned upon completion of normal processing are Copied, Moved, Renamed or Deleted, depending on the function selected.

Other messages which may be returned are:

Message	Explanation
Replaced	The replace option was set to Y and the target error message was deleted before the COPY, MOVE or RENAME function was completed.
Not Replaced	The replace option was set to N and an error message with the same error number already exists in the target environment; function not completed.
Exit: <i>nnn</i>	A user exit routine was active and a non-zero return code was returned by the exit (<i>nnn</i> = the return code); function not completed. See also User Exits.
No Short Err	An extended error message was selected for further processing, but the target error message number has no corresponding short error message; function not completed.
Not Found	An error in the update logic occurred during processing and the requested error message could not be found. This implies that the message was deleted during interim between selection and update.
Ext Exists	The function required a short error message to be deleted, which would have resulted in an extended error message with no corresponding short message; function not completed.
Updated	The text in the specified language did not previously exist for the error number selected, and SYSMAIN has updated the error number with the new language. An error message existed and has now been updated with a new language text.
No Lang 1	Only one language (E or 1) is available for Natural System extended error message text. An attempt has been made to copy extended error message text, and language E or 1 has not been included in the LANGUAGE parameter; function not completed.

Direct Commands for Error Messages

The direct command syntax for processing error messages is shown in this section. (Since the *where-clause* and *with-clause* syntax are identical for each command, they are only shown once with the COPY and MOVE command syntax below.)

For Natural system error messages, Natural-SYSTEM or Natural-SYS has to be specified as library name.

COPY and MOVE Direct Command Syntax

```

{ COPY  
MOVE } ERROR number [THRU number]  

FM [ LIBRARY ] lib-name [where-clause]  

TO [ LIBRARY ] lib-name [where-clause] [with-clause]

```

where-clause

```

[WHERE] [DBID dbid] [FNR file-nr] [NAME name] [CIPHER cipher]  

[ { PASSWORD }  
PSW ] password [LANGUAGE language ] [SEC (dbid, fnr, psw, ciph)]

```

with-clause

```

[WITH] [TYPE type] [REPLACE] [HELP] [RCOP] [MON]

```

Note:

Commas must be used as separators between the values following the SEC keyword; or if a value is missing. For example: SEC (10,,secret,2a). If the Natural session parameter ID has been set to a comma, use a slash (/) sign as the separator between values.

Examples:

```

COPY ERROR 1 FROM ACCOUNTS TO ACCOUNTS1 REP WITH TYPE A  

C ERROR 1 THRU 50 FROM ACCT WHERE DBID 1 FNR 10 LANG 123456  

TO ACCT WHERE DBID 5 FNR 26 LANG 23456 WITH REP HELP

```

```

MOVE ERR 200 THRU 10 FM ACCT FNR 10 LANG 123 TO ACCT LANG 1 TYPE S  

M ERR 376 TYPE E FM ACCT LANG E TO ACCT LANG FGSDI

```

DELETE Direct Command Syntax

```
DELETE ERROR number [THRU number]  
[IN [LIBRARY] lib-name] [where-clause] [with-clause]
```

Examples:

```
DELETE ERROR 1 THRU 10 IN LIBRARY ACCT  
WHERE DBID 1 FNR 2 PSW GUESS CIPH 137561 WITH TYPE E MON HELP  
DEL ERR 100 IN ACCT
```

FIND Direct Command Syntax

```
FIND ERROR number [IN [LIBRARY] lib-name] [where-clause] [with-clause]
```

Examples:

```
FIND ERR 4280 IN A* MON  
FIND ERROR 10 IN LIB ACCT WHERE DBID 1 FNR 3 WITH TYPE E
```

LIST Direct Command Syntax

```
LIST ERROR number [THRU number] [IN [LIBRARY] lib-name]  
[where-clause] [with-clause]
```

Examples:

```
LIST ERR 1 THRU 10 IN ACCT  
LIST ERROR 100 THRU 150 IN LIB ACCT WHERE DBID 12 FNR 5
```

RENAME Direct Command Syntax

```
RENAME ERROR number [THRU number] AS new-number  
[THRU new-number] [with-clause]  
IN [LIBRARY] lib-name [where-clause]  
TO [LIBRARY] lib-name [where-clause]
```

Examples:

```
RENAME ERROR 1 THRU 50 AS 11 THRU 60 WITH TYPE A REP HELP MON RCOP  
IN LIBRARY ACCT WHERE DBID 1 FNR 2 TO LIB ACCOUNT WHERE FNR 3  
RENAME ERR 1 AS 101 IN ACCT
```

Error Messages in Batch Mode

Direct commands must be used to process error messages in batch mode.

A report is provided to show the status of error messages processed in batch mode.

Note:

By using Online Report Mode, you can obtain the SYSMAIN batch report online. If required, you can also obtain a hardcopy of the report using the %H option.

SYSMAIN Profiles

All SYSMAIN functions except the FIND function can be performed on profiles.

The profile environment specification must always correspond to the relevant DBID and FNR of the FNAT file.

This section covers the following topics:

- Profile Menus
- Processing Status
- Direct Commands for Profiles
- Profiles in Batch Mode

Profile Menus

The Profiles menu contains all of SYSMAIN functions for the processing of profiles (there are no subfunction menus for profiles):

```

18:34:24          ***** NATURAL SYSMAIN UTILITY *****          1999-11-25
User SAG                - Profiles -                          Library SYSMAIN

                                Code  Function
                                C    Copy   Profiles
                                D    Delete Profiles
                                L    List   Profiles
                                M    Move   Profiles
                                R    Rename Profiles
                                ?    Help
                                .    Exit

                                Code ..... C                      Sel. List .. Y
Profile Name ..... *_____ Type ... ____ New Name ... _____
Source Database ... 10____ File ... 32____ Name ..... _____
Target Database ... 10____ File ... 32____ Name ..... _____
Options Replace .... N

Command ==>
Enter-PF1---PF2---PF3---PF4---PF5---PF6---PF7---PF8---PF9---PF10--PF11--PF12---
      Help Menu Exit Copy Del          List Move Ren Fsec          Fnat
    
```

If you leave the field Type blank or enter an asterisk (*), editor profiles, map profiles and device profiles will be processed. To process parameter profiles (as created with the SYSPARM utility), you enter a **P**.

If selective processing has been selected, a selection list is displayed. If there are more profiles in a library than can be displayed on a single screen, the additional profiles can be displayed by pressing ENTER.

Select specific profiles for processing by entering the desired options in column **C** (code) to the left of the profile name (the column contains three bytes, and the options can be entered in any order). A question mark (?) in this column invokes a window listing the options available for the specific function. These options make it possible to perform additional functions before actively processing the object.

In a profile selection list, you can enter the following options:

Process All Profiles	If option A is entered, all profile types listed in the corresponding Type column are processed.
Process Editor Profile	Option E is used to process only the editor profile, even if there are map and device profiles listed as well. If E is specified for an object which does not exist as an editor profile, an error is returned.
Process Device Profile	Option D is used to process only the device profile, even if there are editor and map profiles listed as well. If D is specified for an object which does not exist as a device profile, an error occurs.
Process Map Profile	Option M is used to process only the map profile, even if there are editor and device profiles listed as well. If M is specified for an object which does not exist as a map profile, an error is returned.
Process Parameter Profile	Option P is used to process only the parameter profile (as created with the SYSPARM utility).
List Parameter Profile	Option L is used to display the contents of the parameter profile.

Processing Status

After the individual profiles have been selected on the selection list, they are processed by SYSMAIN and an appropriate message is displayed in the message column.

Message	Explanation
Exit: <i>nnn</i>	A user exit routine was active and a non-zero return code was returned by the exit (<i>nnn</i> = the return code); function not completed. See also User Exits.
Dev exists	Replace option was set to N and a device profile already exists in the target environment; all profiles with this name are rejected.
Map exists	Replace option was set to N and a map profile already exists in the target environment; all profiles with this name are rejected.
Edt exists	Replace option was set to N and an edit profile already exists in the target environment; all profiles with this name are rejected.
Replaced	Replace option was set to Y and the target profile was deleted before the COPY, MOVE or RENAME function was completed.
Not Replaced	Replace option was set to N and a profile with the same profile name already exists in the target environment; function not completed.
Not Found	An error in the update logic occurred during processing and the requested profile could not be found; implies that profile was deleted during interim between selection and update.

Direct Commands for Profiles

The direct command syntax for processing of profiles is shown in this section. (Since the *where-clause* and *with-clause* syntax are identical for each command, they are only shown once with the COPY and MOVE command syntax below.)

COPY and MOVE Direct Command Syntax

```

{ COPY
 MOVE } PROFILE name [with-clause] FM [where-clause] TO [where-clause]
    
```

where-clause

```

[WHERE] [DBID dbid] [FNR file-nr] [NAME name]
[CIPHER cipher] [ { PASSWORD
                   PSW } password ]
    
```

with-clause

```

[WITH] [TYPE type] [REPLACE] [HELP] [RCOP] [MON]
    
```

Examples:

```
COPY PROF USER1 TYPE E FM DBID 1 FNR 5 TO DBID 2 FNR 5
COPY PROFILE USER TYPE MED REP FM FNR 6 TO FNR 7
COPY PROF USER1 TYPE P FM DBID 10 FNR 44 TO DBID 3 FNR 7
```

```
MOVE PROF USER1 TYPE E FM DBID 1 FNR 5 TO DBID 2 FNR 5
MOVE PROFILE USER1 TYPE MED REP FM FNR 6 TO FNR 7
```

DELETE Direct Command Syntax

```
DELETE PROFILE name [IN [where-clause]] [with-clause]
```

Examples:

```
DEL PROF U* TYPE DM
DEL PROF TEST* IN DBID 177 FNR 205
```

LIST Direct Command Syntax

```
LIST PROFILE name [IN [where-clause]] [with-clause]
```

Examples:

```
LIST PROF USER* IN DBID 1 FNR 5
LIST PROF DT* TYPE E
```

RENAME Direct Command Syntax

```
RENAME PROFILE name AS new-name
                        [IN where-clause]
                        [TO where-clause] [with-clause]
```

Examples:

```
R PROFILE USER1 AS USER2 RCOP
REN PROF USER1 AS USER2 DBID 1 FNR 4 TO DBID 1 FNR 5
REN PROF USER1 AS NEWUSER IN FNR 4 TO FNR 5 REPLACE RCOP
```

Profiles in Batch Mode

Direct commands must be used to process profiles in batch mode.

A report is provided to show the status of profiles processed in batch mode.

Note:

By using Online Report Mode, you can obtain the SYSMAIN batch report online. If required, you can also obtain a hardcopy of the report using the %H option.

SYSMAIN Rules

All SYSMAIN functions can be performed on processing rules.

The processing rule environment specification must always correspond to the relevant DBID and FNR of the FDIC files.

If Predict is installed, it is recommended that you use Predict instead of SYSMAIN for the processing of rules.

This section covers the following topics:

- Rule Menus
- Processing Status
- Direct Commands for Rules
- Rules in Batch Mode

Rule Menus

The Rules menu contains all of SYSMAIN functions for the processing of rules (there are no subfunction menus for rules):

```

19:12:03          ***** NATURAL SYSMAIN UTILITY *****          1999-11-18
User SAG                - Rules -                                Library SYSMAIN

                                Code  Function
                                C    Copy   Rules
                                D    Delete Rules
                                L    List   Rules
                                M    Move   Rules
                                R    Rename Rules
                                ?    Help
                                .    Exit

Rule      Code ..... C                Sel. List ... Y
Name ..... _____ Type ... ___
New Name .. _____

Source   Database .. 10___ File .. 50___ Name ..... _____
Target   Database .. 10___ File .. 60___ Name ..... _____
Options  Replace ... N

Command ==>
Enter-PF1---PF2---PF3---PF4---PF5---PF6---PF7---PF8---PF9---PF10--PF11--PF12---
Help Menu Exit Copy Del          List Move Ren Fsec Fdic
    
```

The Rule Type is used as part of the selection criteria. Since processing rules are either automatic rules or free rules, only types **A**, **F** or **AF** apply. Only rules which were cataloged with the specified type are selected. With type **AF**, both automatic and free rules are selected. If the Type field is left blank, no verification takes place.

If selective processing has been selected, a selection list is displayed with the following items of information:

Rule Name	The name of the rule that meets the selection criteria and therefore appears on the list.
Type	The type of the selected rule; that is, free rule or automatic rule.
Ver. Type	The verification type is an attribute of the rule which is directly taken from the verification as defined in Predict. A value of Unknown indicates that there is no corresponding verification information in Predict available. For valid values, see the relevant Predict documentation.
Format	The format type is an attribute of the rule which is directly taken from the verification as defined in Predict. A value of Unknown indicates that there is no corresponding verification information in Predict available. For valid values, see the relevant Predict documentation.
Message	The message which indicates the current processing status of the selected rule. See also the following section.

If there are more rules selected than can be displayed on a single screen, the additional rules can be displayed by pressing ENTER.

Processing Status

A help character (?) in column C (code) to the left of the rule name invokes a window that lists the options available for the specific function.

If you want to select a specific rule for processing, enter option A (Process Rule). After the individual rules have been selected on the selection list, they are processed by SYSMAIN and one of the following processing status messages is displayed in the message column.

Message	Explanation
Replaced	Replace option was set to Y and the target rule was deleted before the COPY or MOVE function was completed.
Not Replaced	Replace option was set to N, a rule with the same rule name already exists in the target environment, and no new name has been specified; function not completed.
Err: NAT4852	A Natural Security violation occurred; function not completed.
Exit: <i>nnn</i>	A user exit routine was active and a non-zero return code was returned by the exit (<i>nnn</i> = the return code); function not completed. See also User Exits.

The other options which can be entered on a COPY, DELETE, LIST, MOVE or RENAME selection list have the following meanings:

List DDMs and Fields for this Rule	All DDMs and fields which have the specified rule assigned are displayed. Upon completion of processing the message File Listed is returned. This option only applies to automatic rules.
Hardcopy of Saved Rule	The rule(s) specified are printed and listed on the screen. Upon completion of processing the message Printed is returned. This option only applies to saved rules.
List Directory Information	This option lists the source directory of the specified rule(s). Upon completion of processing the message Directory is returned.
List Saved Rule	The source code of the rule specified is listed. Upon completion of processing the message Listed is returned. This option only applies to saved rules.

Direct Commands for Rules

The direct command syntax for processing of rules is shown in this section. (Since the *where-clause* and *with-clause* syntax are identical for each command, they are only shown once with the COPY and MOVE command syntax below.)

COPY and MOVE Direct Command Syntax

```
{ COPY }
{ MOVE } RULE name [AS new-name] FM [where-clause] TO [where-clause] [with-clause]
```

where-clause

```
[WHERE] [DBID dbid] [FNR file-nr] [NAME name] [CIPHER cipher ]
      { PASSWORD } password
      { PSW }
      [DIC (dbid, fnr, psw, ciph)]
      [SEC (dbid, fnr, psw, ciph)]
```

Note:

Commas must be used as separators between the values following the DIC and SEC keywords; or if a value is missing. For example: DIC(10,,secret,2a). If the Natural session parameter ID has been set to a comma, use a slash (/) sign as the separator between values.

with-clause

```
[WITH] [TYPE type] [REPLACE] [HELP] [MON]
```

Examples:

```
C RULE TESTRULE FM FNR 20 TO FNR 24 REPLACE
COPY RULE C< FM FNR 20 TO FNR 24
```

```
M RULE TESTRULE FM FNR 20 TO FNR 24 REPLACE
MOVE RULE C< FM FNR 20 TO FNR 24
```

DELETE Direct Command Syntax

```
DELETE RULE name [with-clause] [IN [where-clause]]
```

Example:

```
DEL RULE DEMORULE IN DBID 12 FNR 27
```

LIST Direct Command Syntax

```
LIST RULE name [IN [where-clause]] [with-clause]
```

Example:

```
L RULE * DBID 1 FNR 5
```

Rules in Batch Mode

A report is provided to show the status of rules processed in batch mode.

Note:

By using Online Report Mode, you can obtain the SYSMAIN batch report online. If required, you can also obtain a hardcopy of the report using the %H option.

SYSMAIN DDMs

All SYSMAIN functions except the FIND and RENAME functions can be performed on DDMs.

The DDM environment specification must always correspond to the relevant DBID and FNR of the FDIC files.

This section covers the following topics:

- DDM Menus
- Processing Status
- Direct Commands for DDMs
- DDMs in Batch Mode

DDM Menus

The DDMs menu contains all SYSMAIN functions for the processing of DDMs (there are no subfunction menus for DDMs):

```

18:37:11          ***** NATURAL SYSMAIN UTILITY *****          1999-11-18
User SAG                - DDMs -                                Library SYSMAIN

                                Code  Function
                                C    Copy   DDMs
                                D    Delete DDMs
                                L    List   DDMs
                                M    Move   DDMs
                                ?    Help
                                .    Exit

      Code ..... C      Sel. List ... Y
DDM   Name ..... _____
      DDM DBID ... _____ DDM FNR ..... _____
Source Database ... 10___ File ..... 50__ Name ... _____
Target Database.... 10___ File ..... 60__ Name ... _____
Options Replace .... N      Del.NSC-Def. N

Command ==>
Enter-PF1---PF2---PF3---PF4---PF5---PF6---PF7---PF8---PF9---PF10--PF11--PF12---
      Help Menu Exit Copy Del          List Move          Fsec Fdic
    
```

The DBID and FNR of the DDM are used as part of the selection criteria. Only DDMs which were cataloged with the DBID and/or FNR specified are selected. If these fields are set to zero, no verification takes place.

If selective processing has been selected, a selection list is displayed.

If there are more DDMs in a library than can be displayed on a single screen, the additional DDMs can be displayed by pressing ENTER.

Note:

If a DDM is deleted with SYSMAIN, the corresponding Natural Security file profile is automatically deleted, too.

Select specific DDMs for processing by entering option **A** (Process DDM) in column **C** (code) to the left of the DDM name. A question mark (?) in this column invokes a window listing all options available for the specific function.

The options which can be entered on a COPY, DELETE or MOVE selection list have the following meanings:

Process DDM	Process all DDMs specified. Option A is not available with the LIST function.
List DDM	List the specified DDM before performing any action on it.
List Automatic Rules	List all automatic rules linked to the specified DDM(s). If one or more DDMs have been selected for listing of their automatic rules, the message Rules Listed is displayed.

The field Del. NSC-Def. indicates deletion of Natural Security definitions. If a DDM is deleted from a source environment or moved to a new environment, and different FSEC files have been specified, you can use this parameter to specify whether or not to delete the DDM definition in the source FSEC file. Indicate **N** to keep the DDM definition in the source FSEC. Indicate **Y** to delete the DDM definition in the source FSEC.

Note:

This field only appears in a Natural Security environment.

Processing Status

After the individual DDMs have been selected on the selection list, they are processed by SYSMAIN and an appropriate message is displayed in the message column.

Message	Explanation
Replaced	Replace option was set to Y and the target DDM was deleted before the COPY or MOVE function was completed.
Not Replaced	Replace option was set to N and a DDM of the same name already exists in the target environment; function not completed.
Err: NAT4852	A Natural Security violation occurred; function not completed.
Exit: <i>nmn</i>	A user exit routine was active and a non-zero return code was returned by the exit (<i>nmn</i> = the return code); function not completed. See also User Exits.

Direct Commands for DDMs

The direct command syntax for processing DDMs is shown in this section. (Since the *where-clause* and *with-clause* syntax are identical for each command, they are only shown once with the COPY and MOVE command syntax below.)

Note:

For compatibility reasons, instead of the keyword DDM you can use the keyword VIEW (its short form V) in direct commands for DDMs.

COPY and MOVE Direct Command Syntax

```
{COPY}  
{MOVE} DDM name FM [where-clause] TO [where-clause] [with-clause]
```

where-clause

```
[WHERE] [DBID dbid] [FNR file-nr] [NAME name] [CIPHER cipher ]  
      [ {PASSWORD}  
      PSW ] password ]  
[DIC (dbid, fnr, psw, ciph)  
[SEC (dbid, fnr, psw, ciph)
```

Note:

Commas must be used as separators between the values following the DIC and SEC keywords; or if a value is missing. For example: DIC(10,,secret,2a). If the session parameter ID has been set to a comma, use a slash (/) sign as the separator between values.

The DBID, FNR, CIPHER and PASSWORD specifications can be used instead of the corresponding DIC specifications, or vice versa. If an item is specified twice, the one specified last will be used.

with-clause

```
[WITH] [DDMDBID DDM-dbid] [DDMFNR DDM-fnr] [REPLACE] [HELP] [MON]
```

Examples:

```
C DDM PERSONNEL FM FNR 20 TO FNR 24 REPLACE  
COPY DDM C< FM FNR 20 TO FNR 24
```

```
M DDM PERSONNEL FM FNR 20 TO FNR 24 REPLACE  
MOVE DDM C< FM FNR 20 TO FNR 24
```

DELETE Direct Command Syntax

```
DELETE DDM name [with-clause] [IN [where-clause]
```

Example:

```
DEL DDM FINANCE IN DBID 12 FNR 27
```

LIST Direct Command Syntax

```
LIST DDM name [IN [where-clause]] [with-clause]
```

Example:

```
L DDM * IN DBID / FNR 5
```

DDMs in Batch Mode

A report is provided to show the status of DDMs processed in batch mode.

By using Online Report Mode, you can obtain the SYSMAIN batch report online. If required, you can also obtain a hardcopy of the report using the **%H** option.

SYSMAIN DL/I Subfiles

This section covers the following topics:

- General Information
- Processing Status
- Direct Commands for DL/I Subfiles

General Information

If the Natural interface to DL/I is installed, SYSMAIN can also be used to process DL/I subfiles.

Subfiles are:

- **NDBs** = DL/I DBDs (Database Descriptions) defined to Natural.
- **NSBs** = DL/I PSBs (Program Specification Blocks) defined to Natural.
- **UDFs** = DL/I User-Defined Fields defined to Natural.

For more information on NDBs, NSBs, DBDs, PSBs and UDFs, see the Natural for DL/I documentation.

The DL/I Subfiles menu contains all SYSMAIN functions for the processing of DL/I Subfiles (there are no subfunction menus for DL/I Subfiles):

```

11:42:45          ***** NATURAL SYSMAIN UTILITY *****          1999-11-22
User SAG          - DL/I Subfiles -                               Library SYSMAIN

                                Code  Function
                                C    Copy   Subfiles
                                D    Delete Subfiles
                                L    List   Subfiles
                                M    Move   Subfiles
                                ?    Help
                                .    Exit

                                Code ..... C                      Sel. List ... Y

Subfile Name ..... *_____ Type ... D
Source Database .. 10___ File ... 50__ Name ..... _____
Target Database .. 10___ File ... 60__ Name ..... _____
Options Replace ... N

Command ==>
Enter-PF1---PF2---PF3---PF4---PF5---PF6---PF7---PF8---PF9---PF10--PF11--PF12---
      Help Menu Exit Copy Del           List Move           Fsec Fdic
    
```

The SYSMAIN handling of DL/I subfiles is analogous to the handling of DDMs, with the difference that in direct commands you use the keyword DL1 instead of DDM.

In addition, instead of the DDM DBID and FNR, with DL/I subfiles, the subfile type can be specified as part of the selection criteria. Possible types are **D** for NDBs and UDFs, and **P** for NSBs. If the Type field is left blank, an error message is returned.

If selective processing has been selected, a selection list is displayed.

If there are more DL/I subfiles in a library than can be displayed on a single screen, the additional DL/I subfiles can be displayed by pressing ENTER.

Select specific DL/I subfiles for processing by entering option A (see below) in column C (code) to the left of the DL/I subfile name. A question mark (?) in this column invokes a window listing the options available for the specific function.

The options which can be entered on a COPY, DELETE or MOVE selection list have the following meanings:

Option	Explanation
A	Process NSB Subfile or NDB and UDF Subfile Process all DL/I subfiles specified. Option A is not available with the LIST function.
L	List NSB Subfile or NDB and UDF Subfile List the specified DL/I subfile before you perform any action on it.

Processing Status

After the individual DL/I subfiles have been selected on the selection list, they are processed by SYSMAIN and an appropriate message is displayed in the message column.

Message	Explanation
Replaced	Replace option was set to Y and the target DL/I subfile was deleted before the COPY or MOVE function was completed.
Not Replaced	Replace option was set to N and a DL/I subfile with the same subfile name already exists in the target environment; function not completed.
Err: NAT4852	A Natural Security violation occurred; function not completed.
Exit: <i>nnn</i>	A user exit routine was active and a non-zero return code was returned by the exit (<i>nnn</i> = the return code); function not completed. See also User Exits.

Direct Commands for DL/I Subfiles

The direct command syntax for processing DL/I subfiles is shown in this section. (Since the *where-clause* and *with-clause* syntax are identical for each command, they are only shown once with the COPY and MOVE command syntax below.)

COPY and MOVE Direct Command Syntax

<div style="border: 1px solid black; padding: 5px; display: inline-block;"> COPY MOVE </div>	DL1 <i>subfile-name</i> [<i>with-clause</i>] FM [<i>where-clause</i>] TO [<i>where-clause</i>]
--	---

where-clause

```
[WHERE] [DBID dbid] [FNR file-nr] [NAME name]
      [CIPHER cipher] [ {PASSWORD} password ]
                        [PSW
```

with-clause

```
[WITH] [TYPE type] [REPLACE] [HELP] [MON]
```

Examples:

```
COPY DL1 SUBFILE1 TYPE D FM DBID 1 FNR 5 TO DBID 2 FNR 5
COPY DL1 SUBFILE REP FM FNR 6 TO FNR 7 TYPE D
COPY DL1 SUBFILE1 TYPE P TO DBID 3 FNR 7
```

```
MOVE DL1 SUBFILE1 TYPE D FM DBID 1 FNR 5 TO DBID 2 FNR 5
MOVE DL1 SUBFILE1 REP FM FNR 6 TO FNR 7 TYPE D
```

DELETE and LIST Direct Command Syntax

```
{DELETE}
LIST } DL1 subfile-name [IN [where-clause]] [with-clause]
```

Examples:

```
DEL DL1 S* TYPE D
DEL DL1 TEST* IN DBID 177 FNR 205 TYPE D
```

```
LIST DL1 SUBFILE* IN DBID 1 FNR 5 TYPE D
LIST DL1 SF* TYPE P
```

Commands Issued to SYSMAIN

Several commands can be issued to the SYSMAIN utility. These commands perform special functions related to the operation of the utility itself.

Command	Function
ADAON / NOADA	Function to trap abnormal database errors (only applicable online with programming objects) for debugging purposes.
<u>BATCH</u> / <u>NOBATCH</u>	Function to switch the SYSMAIN utility into batch mode, whereby all processing is done as if SYSMAIN was running in batch. NOBATCH switches the SYSMAIN utility back to online mode.
CLEAR	Function to clear the current work area. This function can be useful if a large program is in the work area and the SYSMAIN utility therefore requires a larger ESIZE.
<u>DISPLAY</u>	Display the extended text for the error which has occurred (needs not be preceded by SET).
MON / NOMON	Function to trace the current activity in SYSMAIN. During processing, you are informed as to which object is being read, deleted, updated, added, and whether an error occurs. With programming objects, you are also informed about the action taken with the XREF data. This function is effective only with TP environments which can run in non-conversational mode.
PROMPT / <u>NOPROMPT</u>	Function to enable or disable the SYSMAIN prompts. For example, before any deletion, SYSMAIN prompts you for confirmation. With NOPROMPT, no confirmation screen is displayed.
SET	Function to display a window which explains all special SYSMAIN commands.
SET FDIC	Function whereby the Adabas security information for the Dictionary/Predict system file can be specified. This refers to the Natural parameter FDIC or the keyword DIC in batch mode.
SET FNAT	Function whereby the Adabas security information for the SYSMAIN source and target system files can be specified. This is specified in the <i>where-clause</i> in batch mode.
SET FSEC	Function whereby Adabas security information for the Natural Security system file can be specified. This refers to the Natural parameter FSEC or the keyword SEC in batch mode.
SET PC	Function whereby SYSMAIN verifies whether the device is a personal computer (PC). This setting can be intermittently changed with the %+ and %- terminal commands. SET PC then results in SYSMAIN re-verifying the status of the PC parameter.
STATUS	Function to display the current values of certain SYSMAIN variables that are important for Software AG Support.

Command	Function
<u>TOTAL</u>	<p>Function to verify the actual processing of the last SYSMAIN function executed. The following information is displayed:</p> <p>Read Total number of objects which were actually read, based on the Object Name specification.</p> <p>Rejected Total number of objects read which were then rejected, based on the selection criteria specified.</p> <p>Processed Total number of objects which satisfied the selection criteria.</p> <p>Added Total number of new objects added to the target environment.</p> <p>Updated Total number of existing objects updated. (Where possible, SYSMAIN attempts to update existing objects instead of deleting and adding new ones.)</p> <p>Deleted Total number of objects deleted from either the source or target environment, depending on the function and Replace option.</p> <p>Replaced Total number of objects which were replaced in the target environment.</p> <p>Not Repl. Total number of objects which were not replaced in the target environment.</p>
.	Terminates SYSMAIN (SET is not required).

SYSMAIN PF Keys

You can also use PF keys to perform SYSMAIN functions. PF keys which are not valid within certain menus, are not displayed for this menu. The PF keys are summarized in the table below.

PF Key	Name	Function
PF1	Help	Displays online help depending on current cursor position. If the cursor is positioned in the Object Code or Function Code field, SYSMAIN general help is displayed. If the cursor is in another field, field-specific help is displayed.
PF2	Menu	Displays the SYSMAIN Main Menu.
PF3	Exit	Returns to the previous screen. If you press PF3 on the SYSMAIN Main Menu, SYSMAIN is terminated.
PF4	Copy	Performs the COPY function for the object specified.
PF5	Del	Performs the DELETE function for the object specified.
PF6	Find	Performs the FIND function for the object specified. (Only with programming objects, error messages and rules.)
PF7	List	Performs the LIST function for the object specified.
PF8	Move	Performs the MOVE function for the object specified.
PF9	Ren	Performs the RENAME function for the object specified.
PF10	Fsec	Invokes the security screen for specifying the security information of the FSEC file to Natural Security.
PF11	Fdic	Invokes the security screen for specifying the dictionary information of the Natural FDIC file. (Only applicable for programming objects and for rules and DDMs.)
PF12	Fnat	Invokes the security screen for specifying the system file information of the Natural FNAT and/or FUSER files. (Only applicable for programming objects, debug environments, error message texts and profiles.)
CLEAR	Exit	Returns to the previous screen. If you press CLEAR on the SYSMAIN Main Menu, the SYSMAIN utility is terminated.

SYSMAIN - Data Rejected

This section covers the following topics:

- Object Rejection and Reasons
 - SYSMAIN Error Notification
-

Object Rejection and Reasons

If, during the execution of a SYSMAIN function, one or more objects were found to satisfy the specified selection criteria, but some or all of these objects were then rejected for further processing, an error or NAT4893 occurs. The possible reason for these errors are:

- An object was selected and then rejected because the object type was not valid for the type of processing specified. For example, all maps are rejected if processing is for programs or subroutines.
- An object was selected and then rejected because the date on which it was saved or cataloged did not fall within the range specified by the Date/Time From, Date/Time To, User ID and Terminal ID parameters.
- A cataloged programming object with type **S** (subroutine) was selected and then rejected because the external name was identical to the name of another subroutine in the target library.
- A cataloged programming object with type **4** (class) was selected and then rejected because the external name or the GUID was identical to the name or GUID of another class in the target library.
- An object was selected and then rejected because the target environment already contained an object identified by the same name, and the Replace option was set to **N**.
- A cataloged object was selected and then rejected because the Recat option was **ON** and there was no saved object corresponding to the cataloged object.
- A saved (only) object was selected and then rejected because the Recat option was **ON** and the target environment already contained a cataloged object with the same name.
- The XREF indicator was not set to **N** and there were no XREF data for the programming object specified.
- A user exit routine was active, and a non-zero return code was returned during processing of the object.
- An error message extended text was selected and then rejected because there was no corresponding short message text in the source library.
- An extended error message text was selected but could not be processed because there was no short error message text in the target environment.
- A short error message text was selected to be moved, deleted or renamed, but could not be processed because the corresponding extended error text was not included in the selection criteria. An extended error message must always have a corresponding short error message text.
- The library is controlled by PAC/PAA, and the object can be handled by using the NATLOAD utility only.
- A protected library under the control of Natural Security includes restrictions on object types, which are therefore ignored by SYSMAIN.

You can use the SYSMAIN utility command **TOTAL** to review the specific status of a request.

SYSMAIN Error Notification

SYSMAIN always attempts to recover in the event of an error during processing. This feature is automatically activated and uses the Natural system variable ***ERROR-TA**. This feature is deactivated when SYSMAIN is terminated normally.

If the terminal command control character is used to terminate SYSMAIN, this is considered an abnormal termination, and the system variable ***ERROR-TA** is not reset. It can be reset by re-invoking SYSMAIN and terminating it normally. In the event that you have set the ***ERROR-TA** system variable, SYSMAIN resets it to its previously assigned value upon termination.

If invalid data have been specified with respect to the selection criteria, an error message is displayed in the message line. If you are uncertain as to the meaning of the short error message, the SYSMAIN command DISPLAY, DISP or DIS can be entered to activate a display of the corresponding extended error message text.

Data Entry Errors

If invalid data have been specified with respect to the selection criteria, an error message is displayed in the message line. In some situations the online help facility for particular entries is invoked. This feature provides you with more detailed information about the error.

If an error occurs in batch mode, an error message and corresponding error number are printed and the SYSMAIN utility is terminated.

Processing Errors

If you make a request which causes an error in processing, the SYSMAIN utility ERROR module is invoked, and a window is displayed:

```

11:29:36          *** SYSMAIN Error Report ***          1999-11-25

The following internal error occurred while processing the
SYSMAIN function: XXXXXX (CC)

    Error in field specification for IF SELECTION statement.

Error Number .. EEEE
Program ..... PPPPPPPP
Status Code ... S           Status ..... Unknown Error Type
Line ..... LLLL           Level ..... VV
Library ..... SYSMAIN     Steplib .....
Startup .....             Device ..... PC
User Id ..... SAG         User Name ... USER1
    
```

The information contained in the window is useful for analyzing the cause of the error.

The values in the window above have the following meanings:

<i>XXXXXX</i>	SYSMAIN function.
<i>CC</i>	An internal status code useful for Software AG support personnel. The following codes can be displayed: A Automatic processing D XREF data are being deleted E Error in processing (flag for SYSMAIN) F Status setting when XREF data are being processed G Status setting when XREF data are being processed H Selection list processing I Option is being processed S Single object processing T SYSMAIN termination by command processor V Status setting when XREF data are being processed X SYSMAIN termination by command processor Y Validation error has occurred, redisplay should follow Z Validation error has occurred, redisplay should follow
<i>EEEE</i>	Corresponds to the system variable *ERROR-NR.
<i>LLLL</i>	Corresponds to the system variable *ERROR-LINE.
<i>S</i>	C Command processing error. L Logon error. O Object time error. S Non-correctable Syntax error.
<i>VV</i>	Corresponds to the system variable *LEVEL.
<i>PPPPPPP</i>	Corresponds to the system variable *PROGRAM.

If a processing error occurs, note the information in the window and press ENTER. The SYSMAIN utility attempts to recover to the last active menu screen, leaving the data values of the parameters unchanged.

If DISPLAY, DISP or DIS is entered in the window, the extended error message for the error incurred is displayed.

If a processing error occurs during batch processing, the SYSMAIN utility prints the relevant error message and terminates.

Certain user errors can also cause the window to be displayed. Although SYSMAIN attempts to trap all errors during evaluation, this may not always be entirely successful. For example, if a user requests that a DDM be copied from one environment to another, but specifies an invalid DBID, SYSMAIN attempts to access this database. An Adabas response code of 148 is returned, and the SYSMAIN ERROR module is invoked and the window displayed. Similarly, an invalid file can result in a number of errors being sent from the database.

In situations in which an Adabas response code 9 is returned, SYSMAIN writes a message informing you of the error and restart processing from the last function or subfunction menu. If a particular request had not been completed, you can assume that the response code 9 resulted in a BACKOUT TRANSACTION to the last non-completed transaction.

Special Considerations for Administrators

This section covers the following topics:

- SYSMAIN Security
- User Exits

SYSMAIN Security

The security aspects of the SYSMAIN utility can be divided into two categories:

- File Security
- Natural Security

File Security

The file security relates to the security which users can define for their Natural system files, FUSER, FNAT, FDIC and FSEC. This type of security can be used in both Adabas and VSAM environments.

To define file security to the SYSMAIN utility (that is, passwords and cipher codes), invoke the SYSMAIN utility security screens and enter the appropriate security profiles. The security screens are invoked as shown in the following section:

File	Special Command	Key	Objects Affected
FUSER, FNAT	SET FNAT	PF12	programming objects debug environments error messages profiles
FDIC	SET FDIC	PF11	rules DDMs XREF information DL/I subfiles
FSEC	SET FSEC	PF10	Natural Security profile

The security screens are windows similar to the one shown below:

```
--- Security for the NATURAL System Files ---

Specify the password(s), cipher(s) and VSAM FCT
name(s) for the source/target file(s) below:

    - Source -                - Target -
Library .... OLDLIB          Library .... NEWLIB
Database ... 10              Database ... 10
File ..... 32                File ..... 51

Password ...                 Password ...
Cipher .....                 Cipher .....
VSAM Name .. _____     VSAM Name .. _____
```

The FUSER and FNAT files relate to the source and target environments since these environments must relate to the appropriate Natural system file.

Note:

No validation is performed for any of the information. The field or entry Library is applicable only when processing programming objects.

If the security information is not provided and the Natural system file requires security, the Adabas security for the appropriate Natural system file is taken as the default.

You can override this feature by invoking the appropriate security screen as mentioned above. Once file security is defined, the SYSMAIN utility uses this security information for all subsequent processing. If a user then requires that the default security information (obtained at initialization of the session) be used, the corresponding security screen must be re-invoked and the password and cipher code fields set to blank. (The password and cipher codes are non-display, so even though they appear to be blank, they should be set to blank again.)

Natural Security

Two aspects must be considered when using the SYSMAIN utility within a Natural Security environment:

- Defining the Natural Security Environment
- Restricting Use of SYSMAIN under Natural Security

Defining the Natural Security Environment

The source and target libraries can be within one Natural Security environment or within two different Natural Security environments. These environments must be defined to the SYSMAIN utility.

The definition of the Natural Security environment(s) to be used is specified with the command SET FSEC.

By default the current FSEC setting assigned at the start of the Natural session is used. If you change these settings (on the screen Security for Natural Security), they remain in effect until they are changed by the next SET FSEC process. In batch mode, the SEC parameter should be used to specify the file security and assignments of the request.

Once the source and target environments have been determined, SYSMAIN verifies both the source and target libraries with Natural Security. (The source and/or target database and file must correspond with the DBID and FNR specified in the library security profile; if these values are not specified, default values are taken from the security profile.)

Restricting Use of SYSMAIN under Natural Security

The use of the SYSMAIN utility itself can be restricted, or the use of the SOURCE and TARGET libraries to be handled with the SYSMAIN utility can be restricted. See Protecting Natural Utilities in the Natural Security documentation for details.

User Exits

The user exits of the SYSMAIN utility provide information about each object being processed.

The user exits are Natural subprograms invoked with CALLNAT statements. All subprograms (as well as the data areas they use) are provided in source form in library SYSMAIN.

The source codes of the subprograms are stored under the names SM-UX-*nn* (*nn* = 01 to 11) in library SYSMAIN. To make a user exit available, you have to stow the corresponding source under the name MAINEX*nn*, either in the library SYSMAIN or in one of its steplibs.

Note:

The names of the user exits' sources and objects are different to ensure that the overwriting of the sources by an update installation does not affect the objects.

You can change or expand any of the user exits as necessary.

Use of these exits results in additional overhead to the SYSMAIN utility, depending on the code logic. It is necessary, however, always to return control to SYSMAIN when exit processing is completed.

As the SYSMAIN utility uses ET logic with Adabas files, the use of user exits can lengthen the transaction time limit (Adabas parameter TT). Furthermore, the definition of the Adabas transaction should not be altered, which means that you should not issue any ET/BT commands or END/BACKOUT TRANSACTION statements. SYSMAIN is responsible for the issuing of all END TRANSACTION statements. The exception to this rule is in a situation where a user terminates the normal completion of any SYSMAIN function with the user exits. If this is the case you **must** issue a BACKOUT TRANSACTION before terminating.

If the return code is set to a non-zero value, this overrides any error given by SYSMAIN. (See the sections relating to specific object types for a discussion of message field settings.) When an error is received from an exit, it is placed in the message field and displayed or printed as appropriate. The exception is automated processing, because processing is completed with minimum terminal I/O.

The individual user exits are described below:

- MAINEX01 - First User Exit for Object Interrogation
- MAINEX02 - Second User Exit for Object Interrogation
- MAINEX03 - User Exit for Request Interrogation
- MAINEX04 - User Exit for Modification of File Assignments
- MAINEX05 - User Exit for Verification of Direct Commands
- MAINEX06 - User Exit for SYSMAIN Initialization
- MAINEX07 - User Exit for SYSMAIN Termination
- MAINEX08 - User Exit for Nothing Found in Batch Mode
- MAINEX09 - User Exit for Abnormal Termination in Batch Mode
- MAINEX10 - User Exit for Command Errors in Batch Mode
- MAINEX11 - User Exit for Setting Special Flags to SYSMAIN

MAINEX01 - First User Exit for Object Interrogation

Function	Interrogate the current value settings of the data elements associated with an object before the object is processed by SYSMAIN.
Remarks	Any object passed to MAINEX01 can be rejected by setting the RESP-CODE parameter to a non-zero value. If any additional logic is to be performed, the transaction may not be at end-of-transaction status and so no END TRANSACTION or BACKOUT TRANSACTION statement should be issued. Control must be returned to SYSMAIN.
Parameters	PARM-AREA1 (A250) /* SYSMAIN parameter area (fixed values) PARM-AREA2 (A250) /* SYSMAIN parameter area (variable values) RESP-CODE (B1) /* Response code to be returned to SYSMAIN Note: Only the RESP-CODE parameter can be modified.
Local Data Area	SM-UX-L

MAINEX02 - Second User Exit for Object Interrogation

Function	Interrogate the current value settings of the data elements associated with an object after the object has been processed by SYSMAIN.
Remarks	Any object passed to MAINEX02 can be rejected by setting the RESP-CODE parameter to a non-zero value. If any additional logic is to be done, the transaction may not be at end-of-transaction status and so no END TRANSACTION or BACKOUT TRANSACTION statement should be issued. Control must be returned to SYSMAIN.
Parameters	PARM-AREA1 (A250) /* SYSMAIN parameter area (fixed values) PARM-AREA2 (A250) /* SYSMAIN parameter area (variable values) RESP-CODE (B1) /* Response code to be returned to SYSMAIN Note: Only the RESP-CODE parameter can be modified.
Local Data Area	SM-UX-L

MAINEX03 - User Exit for Request Interrogation

Function	Interrogate any request made to SYSMAIN in terms of a direct command or information entered via the online menu system. MAINEX03 obtains control before SYSMAIN processes the command.
Remarks	Any command passed to MAINEX03 can be rejected by setting the RESP-CODE parameter to a non-zero value. Additional logic can be added, but it is your responsibility to issue any necessary END TRANSACTION requests to the database. Control must be returned to SYSMAIN.
Parameters	<p>PARM-AREA (A250) /* Command string passed in three parts RESP-CODE (B1) /* Response code to be returned to SYSMAIN</p> <p>Note: Only the RESP-CODE parameter can be modified.</p>

MAINEX04 - User Exit for Modification of File Assignments

Function	Override the database, file, password and cipher codes for the Natural system file(s).
Remarks	MAINEX04 is invoked before any request is processed or validated by SYSMAIN. When control is passed to MAINEX04, you are at end-of-transaction status; therefore you have to set the RESP-CODE parameter to a non-zero value if you wish to reject the request. Control must be returned to SYSMAIN.
Parameters	<p>PARM-AREA (A250) /* SYSMAIN parameter area RESP-CODE (B1) /* Response code to be returned to SYSMAIN</p>
Local Data Area	SM-UX-L4

MAINEX05 - User Exit for Verification of Direct Commands

Function	Verify any direct command entered during online processing of SYSMAIN. In addition, the special characters used to indicate a system command can be overwritten.
Remarks	<p>MAINEX05 is invoked before any direct command issued within SYSMAIN is processed. For example, MAINEX05 enables you to interrogate any of the special SET commands and also prevent them from being issued. You can verify these commands and reject them by returning a non-zero value in the RESP-CODE parameter. You are at end-of-transaction status when control is passed to MAINEX05.</p> <p>A Natural system command entered within SYSMAIN has to be preceded by two slashes (//); see Command Line. With MAINEX05, you can define two other special characters for this purpose; to do so, you assign the desired characters to the parameter CMD-DEL. If CMD-DEL is set to blanks, SYSMAIN uses the default value of two slashes (//). Control must be returned to SYSMAIN.</p>
Parameters	<p>COMMAND (A68) /* Actual command issued in SYSMAIN CMD-DEL (A3) /* Special character for system commands RESP-CODE (B1) /* Response code to be returned to SYSMAIN</p>

MAINEX06 - User Exit for SYSMAIN Initialization

Function	Obtain control at initialization of a SYSMAIN session.
Remarks	MAINEX06 is invoked at the start of the SYSMAIN session, where you can override some of the SYSMAIN default settings, as for example, prompts for confirmation of a request like deleting, moving or replacing an object. All parameters are verified. If they are invalid, the default settings are used. Control must be returned to SYSMAIN.
Parameters	MAINEX06 uses the parameter data area SM-UX-L6.

MAINEX07 - User Exit for SYSMAIN Termination

Function	Obtain control at termination of a SYSMAIN session.
Remarks	MAINEX07 is invoked at termination of a SYSMAIN session to decide whether control is to be kept by SYSMAIN or not.
Parameters	USER-AREA (A50) /* Area for free usage

MAINEX08 - User Exit for Nothing Found in Batch Mode

Function	Determine further processing if no objects are found for a command in batch mode.
Remarks	MAINEX08 is invoked if no objects are found that meet the specified criteria for a specific command executed in batch mode. If this is the case, control may, but need not, be returned to SYSMAIN. If control is returned to SYSMAIN, SYSMAIN will continue processing with the next command.
Parameters	CMD (A250) /* Command string passed in three parts

MAINEX09 - User Exit for Abnormal Termination in Batch Mode

Function	Determine action to be taken in case of error in batch mode.
Remarks	MAINEX09 is invoked if SYSMAIN processing in batch mode leads to an error. If this is the case, control may, but need not, be returned to SYSMAIN. If control is returned to SYSMAIN, SYSMAIN will be terminated with condition code 45. Note: Errors NAT4810, NAT4818, NAT4867, NAT4868 and NAT4893 cannot be handled by this user exit.
Parameters	CMD (A250) /* Command string passed in three parts ERROR-CODE (N4) /* Number of error which caused termination

MAINEX10 - User Exit for Command Errors in Batch Mode

Function	Determine action to be taken in case of command error in batch mode.
Remarks	MAINEX10 is invoked if an error is detected in a SYSMAIN command in batch mode. If this is the case, control may, but need not, be returned to SYSMAIN. If control is returned to SYSMAIN, SYSMAIN will continue processing with the next command.
Parameters	CMD (A250) /* Command string passed in three parts ERROR-CODE (N4) /* Number of error which caused termination

MAINEX11 - User Exit for Setting Special Flags to SYSMAIN

Function	Special settings User Exit.
Remarks	MAINEX11 is invoked at the start of the SYSMAIN session, where you can set some special SYSMAIN flags, as for example, display of MAINUSER messages in batch. See the source of the user exit (SM-UX-11) for the available flags. Control must be returned to SYSMAIN.
Parameters	FLAGS (A250) /*Flag string (redefined)

SYSPARM Utility

The utility SYSPARM is used to create and maintain sets of Natural profile parameters.

When invoking Natural with dynamic profile parameters, you can specify individual parameters each time you invoke Natural. More comfortably, however, you can specify a set of parameters once in SYSPARM, store this set under a profile name, and then invoke Natural with only one dynamic parameter: `PROFILE=profile-name`. The parameters defined in this profile are then passed to Natural as dynamic parameters.

A profile in this context means a set of profile parameters stored under a profile name.

The profiles are stored in the system file FNAT. The database ID (DBID) and file number (FNR) of the current FNAT file are shown on the SYSPARM Menu.

For further details on the profile parameter PROFILE, as well as descriptions of the individual profile parameters that can be specified in a profile, refer to Profile Parameters in the Natural Parameter Reference documentation.

To restrict the use of a profile to specific users, you use the profile parameter USER (as described in the Natural Parameter Reference documentation).

This section covers the following topics:

- Invoking SYSPARM
- List Profiles
- Display Profile
- Add New Profile
- Modify Profile
- Copy Profile
- Delete Profile

Invoking SYSPARM

To invoke the SYSPARM utility

- In the direct command line, enter SYSPARM.
The SYSPARM Menu is displayed.

The SYSPARM Menu provides the following functions:

Function	Explanation
List Profiles	Displays a list of all profiles. From the list, you can select a profile for display, modification or deletion.
Display Profile	Displays a specific profile.
Add New Profile	Creates a new profile.
Modify Profile	Changes an existing profile.
Copy Profile	Creates a new profile by copying an existing one.
Delete Profile	Deletes an existing profile.

List Profiles

This function displays a list of all existing profiles.

To invoke the List Profiles function

- On the SYSPARM Menu, enter Function Code **L** and,
 - Enter an asterisk (*) to list all profiles.

Or, specify a profile name with asterisk (*) notation to list only a certain range of profiles.

The List Profiles screen is displayed.

On the List Profiles screen, press PF7 or PF8 to scroll the list up or down.

Listed below are the PF keys and line commands available to select a profile for display, modification or deletion. Mark a profile with the cursor and press the appropriate PF key, or enter a line command in the Sel. column next to the profile and press ENTER:

Key	Line Command	Function
PF4	D	Invokes the Display Profile function for the profile specified.
PF5	M	Invokes the Modify Profile function for the profile specified.
PF6	X	Invokes the Delete Profile function for the profile specified.

Display Profile

This function is used to display a specific profile.

To invoke the Display Profile function

- On the SYSPARM Menu, enter Function Code **D** and the name of a profile.
Or, from the List Profiles screen, select a profile as described in the relevant section.

The Display Profile screen appears displaying the profile specified.

To modify the profile, press PF5 which invokes the Modify Profile function.

Add New Profile

This function is used to create a new profile.

To invoke the Add New Profile function

- On the SYSPARM Menu, enter Function Code **A** and the name of a profile.

On the screen displayed, specify the profile parameters to be included in the profile as described below:

- Specifying Parameters
- Functions
- Help on Parameter

Specifying Parameters

The individual parameters must be separated from one another by (one or more) blanks or commas.

You can spread the specification of the parameters over as many lines as you like.

When the profile is used with a PROFILE parameter (see the Natural Parameter Reference documentation), the entire specification (including any blanks) is evaluated as one continuous string of dynamic profile parameters - and must therefore conform to the syntax for dynamic parameters.

Note:

The profile can itself contain a PROFILE parameter, which invokes another profile, whose parameter string is then evaluated as part of the string of the first profile.

Functions

The following functions are available on the profile editing screen:

Key	Function
PF4	Checks if the parameter specifications within the profile are syntactically correct.
PF5	Stores the profile.
PF9	Inserts one blank line below the line in which the cursor is positioned.
PF10	Deletes the line in which the cursor is positioned.
PF11	Copies the line in which the cursor is positioned.

Help for Parameter

To invoke help information on a specific profile parameter

- In the field Help for Parameter:
 - Enter the name of a parameter.
 - Or, enter a character string with asterisk (*) notation.
All parameters whose names begin with that character string are listed.
Select a parameter by marking it with any character.
 - Or, enter an asterisk (*).
A list of all parameters is displayed.
Select a parameter by marking it with any character.
- Press ENTER.
Or, place the cursor at a certain editor line and press ENTER as described under Special Positioning below.

The screen that appears contains two sections:

- The upper section with help text on the parameter specified.
If the text extends beyond the current screen, in the field More Help, enter a plus (+) sign to display the next screen.
Enter a minus (-) sign to return to the previous screen or enter a period (.) to terminate the Help for Parameter function.
- The lower section "Enter your parameter specification".
- Under "Enter your parameter specification", enter a parameter specification.
- Press PF4 to have your parameter specifications checked.
- Press PF3 to terminate the Help for Parameter function and add the parameter specifications at the bottom of your profile or at the position marked earlier: see Special Positioning below.

Special Positioning

To add new parameter specifications at a certain position in the profile:

- In the field Help for Parameter, enter a value (or an asterisk) and do **not** press ENTER.
- Position the cursor at the editor line below which you want the new parameter specifications to be placed.
- Press ENTER to invoke the Help for Parameter function and enter a parameter specification.
- Press PF3.

The new parameter specification was added below the editor line marked earlier.

Modify Profile

This function is used to change an existing profile.

To invoke the Modify Profile function

- On the SYSPARM Menu, enter Function Code **M** and the name of a profile.
Or, on the Display Profile screen, press PF5.
Or, from the List Profiles screen, select a profile as described in the relevant section.

A screen (similar to the screen for adding new profiles) is displayed on which you can modify the profile selected.

Copy Profile

This function is used to create a new profile by copying an existing one.

To invoke the Copy Profile function

- On the SYSPARM Menu:
 - Enter Function Code **C**.
 - In the field Profile, enter the name of an existing profile.
 - In the field Copy To, enter the name of the new profile.

Note:

With this function, you can copy profiles only within the same FNAT file. To copy a profile to another FNAT file, use the utility SYSMAIN as described in the relevant documentation.

Delete Profile

This function is used to delete an existing profile.

To invoke the Delete Profile function

- On the SYSPARM Menu, enter Function Code **X**.
Or, from the List Profiles screen, select a profile as described in the relevant section.

The Delete a Profile window is displayed.

- In the Delete a Profile window:
 - Confirm the deletion by entering the name of the profile.
 - Or, press ENTER without entering anything to cancel the action.

Debugging and Monitoring - Overview

Natural offers various utilities for debugging, online testing and monitoring. These utilities allow you to locate errors and test or control programming objects and environments.

This section covers the following topics:

- **Natural Debugger** Is used to test various aspects of a Natural application and assists in locating errors in the processing flow of a program. Provides statistics on objects and statement lines invoked during program execution.
- **DBLOG** Logs database calls: indicates which Adabas commands, DL/I calls or SQL statements are issued by a Natural program.
- **ADACALL (SYSADA)** Issues Adabas direct calls (native commands) directly to an Adabas database.
- **SYSBPM** Monitors and controls the Natural buffer pool.
- **SYSEDT** Displays parameters and runtime information for the editor buffer pool. Modifies parameters and deletes logical work and recovery files.
- **SYSRDC** Collects monitoring and accounting data about the processing flow of a Natural application.
- **SYSTP** Monitors and controls various TP-monitor-specific characteristics of Natural.
- **NATPAGE - Screen Paging** Records screens invoked during a Natural session.
- **Natural Recording** Records commands and input data entered during a Natural session. Re-executes a recorded session.
- **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. See the system command **DUMP** in the Natural Command Reference documentation.

Natural Debugger - Overview

The Natural Debugger, which is part of the Natural Test Utilities, can be used to locate errors in a Natural program.

This section covers the following topics:

- Concepts of the Natural Debugger
- Start the Natural Debugger
- Set Test Mode ON/OFF
- Debug Environment Maintenance
- Spy Maintenance
- Breakpoint Maintenance
- Watchpoint Maintenance
- Call Statistics Maintenance
- Statement Execution Statistics Maintenance
- Variable Maintenance
- List Object Source
- Execution Control Commands
- Navigation and Information Commands
- Command Summary and Syntax

Concepts of the Natural Debugger

The Natural Debugger can be used to temporarily take over control of a Natural session for debugging purposes while a Natural program is executing.

The execution flow of an application is not influenced by the Natural Debugger being applied to it; that is, the application itself need not be adapted to or prepared for its being debugged. By setting debug entries in a program, you can follow the processing flow of the program. The Natural Debugger receives control at any debug entry set, thus allowing various program investigations. This helps you to understand poorly written or poorly documented programs, and to identify redundant or dead (never gets executed) program code. This can be useful, for example, when you take over an application from another developer to maintain, improve or expand it.

In addition, when you write an interface to a program, you can use the Natural Debugger to check the contents of variables at the point when parameters are passed between the program and the interface and thus make sure that parameters are passed correctly.

When Test Mode is set to ON (see Set Test Mode ON/OFF), the Natural Debugger receives control whenever a Natural error occurs during the execution of a program, irrespective of any debug entries defined. You can then, for example, review the contents of the variables in the program to determine the reason for the error.

Below is information on:

- Debug Entries/Spies
 - Debug Window
-

Debug Entries/Spies

Debug entries are also referred to as spies in the Natural Debugger environment. Two types of debug entries (spies) are available: breakpoints and watchpoints. Debug entries for the current debug session can be set, modified, listed, displayed, activated, deactivated and deleted. Debug entries can also be saved for future use as described in Debug Environment Maintenance.

When a debug entry is set or modified, Natural internally stores the library, database ID and file number where the object is located. The object may be located in the current library or in one of its steplibs. If an object of the same name is later executed from another library, the corresponding debug entry is not executed.

The Natural Debugger assigns a name and a unique number (Spy Number) to each debug entry. The name assigned to a debug entry (also referred to as Spy Name) can be either a name specified by the user or a default name created by the Natural Debugger. A debug entry can be selected by its number with the corresponding Natural Debugger commands. If more than one debug entry has to be executed at a specific statement line, they are executed in ascending order of their numbers.

Each debug entry has an initial state and a current state. Possible values are **A** (active) and **I** (inactive). The initial value is specified when setting or modifying the breakpoint or watchpoint and determines the state of the debug entry at environment start or after reset. During the debug session, the state can be changed with the debug commands **ACTIVATE** and **DEACTIVATE** (see also the syntax diagrams in Command Summary and Syntax).

Each debug entry has an event count, which is increased every time the debug entry is executed. A debug entry is not executed if the current state is "inactive". The event count of the breakpoint or watchpoint is not increased either.

The number of executions of a debug entry can be restricted in two ways:

- A number of skips can be specified before the debug entry is executed. The debug entry is then ignored until the event count is higher than the number of skips specified.
- A maximum number of executions can be specified, so that the debug entry is ignored, as soon as the event count exceeds the specified number of executions.

For each debug entry (breakpoint or watchpoint), up to six debug commands can be specified. These commands are executed at execution time of the breakpoint or watchpoint. You can use all Natural Debugger commands that can be applied during a debug interrupt. The default command is the BREAK command, which displays the Debug Window, as shown below.

Attention:

If you delete the BREAK command when setting a debug entry and you do not enter any command that issues a dialog, there is no way to assume control during program interruption.

Debug Window

When the Natural Debugger receives control of the session, the Debug Window is displayed.

The Debug Window shows the type and name of the debug entry that has caused the break (that is, the name of the corresponding breakpoint or watchpoint), its source-code line number, and the name of the interrupted object. For a detailed description of the functions listed below see Execution Control Commands.

From the Debug Window, you can select the following functions:

Function	Code	Description
Go	G	Continues the execution of the program up to the next debug entry specified.
List Break	L	Lists the program code currently active. The last statement executed is highlighted.
Debug Main Menu	M	Invokes the Debug Main Menu which provides all functions needed to maintain debug entries at which control is to be assumed.
Next	N	Executes the next command specified for the current breakpoint or watchpoint.
Run	R	Continues the program with test mode OFF.
Step	S	Continues the program in step mode.
Variable Function	V	Displays the program variables currently active and modifies the contents of these variables.

Start the Natural Debugger

Below is a rough guideline on how to proceed when planning to apply the Natural Debugger. Before starting the Natural Debugger, note the following:

The Natural Debugger can be applied to stowed or cataloged Natural objects only.

The Natural Debugger can only be applied to Natural Version 2.3 or 3.1 objects, but not to objects stowed with any previous version.

Below is information on:

- Invoke the Natural Debugger
 - Default Object
-

Invoke the Natural Debugger

To invoke the Natural Debugger

1. Establish a debug environment for a Natural program or application:
 - Invoke the Debug Main Menu by entering the direct command `TEST`.
Or, from within a running application, enter the terminal command `%<TEST`.
The Debug Main Menu provides the following functions:
 - Set Test Mode ON/OFF
 - Debug Environment Maintenance
 - Spy Maintenance
 - Breakpoint Maintenance
 - Watchpoint Maintenance
 - Call Statistics Maintenance
 - Statement Execution Statistics Maintenance
 - Variable Maintenance
 - List Object Source
2. Use the functions of the Debug Main Menu to specify debug entries for a Natural program or application.
3. Activate the Natural Debugger:
 - In the direct command line or in the NEXT line, enter `TEST ON`.
Or, on the Debug Main Menu, enter Function Code `T`.
4. Execute the Natural program or application.

Default Object

The maintenance functions of the Natural Debugger as described in the relevant sections, refer to objects you specify either in the corresponding name fields of menus or with direct commands. If you do not specify an object name, by default, the Natural Debugger assumes the name of the current object as it is displayed in the field Object, in the upper right corner of the screen. With a default object defined, no object name is required in direct commands and menu options used to specify breakpoints or watchpoints. To change the default object, see the syntax of the command SET in the section Command Summary and Syntax.

Set Test Mode ON/OFF

To activate a previously established debug environment, test mode must be switched on.

 **To set Test Mode to ON or OFF**

- On the Debug Main Menu enter Function Code **T** to activate and deactivate.
Or enter the direct command **TEST ON** or **TEST OFF**.

When starting a program with test mode **ON** or when activating test mode while running a program, the Natural debugger continuously checks all debug entries for any required action.

When starting a program with test mode **OFF** or when test mode is deactivated while running a program, all debug entries are ignored.

The command **TEST**, and with it the whole application, can be protected by Natural Security. Individual functions within the Natural Debugger, however, cannot be protected.

Debug Environment Maintenance

Since a debug environment mainly consists of debug entries, it is established by setting breakpoints and watchpoints, as described in the relevant maintenance sections.

Once established, a debug environment can be stored for subsequent usage. The file where debug environments are stored can be set with the direct command PROFILE (see Navigation and Information Commands). You can also delete a debug environment or reset its counters to their initial values.

The following items are also part of a debug environment and are therefore saved or loaded every time you save or load a debug environment:

- Test Mode setting (ON/OFF);
- all options that can be set with the direct command PROFILE except the file for loading or saving debug environments;
- the settings of the statement execution statistics (ON/OFF/COUNT).

You can perform all these functions using the Debug Environment Maintenance menu, which is invoked by entering Function Code **E** on the Debug Main Menu.

Note:

The Natural Debugger can only maintain debug environments created with Natural Version 2.3 or 3.1; debug environments created with any previous version will be ignored.

The functions provided with the Debug Environment Maintenance menu are listed below. With each function selected, you must enter the name of the debug environment to be maintained:

- Set Test Mode ON/OFF
(see the relevant section)
 - Load Debug Environment
 - Save Debug Environment
 - Reset Debug Environment
 - Delete Debug Environment
-

Load Debug Environment

To load a debug environment from your user system file (FUSER)

- On the Debug Environment Maintenance menu, enter Function Code **L** and the name of an environment. Or enter the direct command LOAD ENVIRONMENT *name*. The specified debug environment is loaded.

If you do not specify a name, the default environment with the name Noname is loaded.

Enter an asterisk (*) to obtain a list all available debug environments. On the list, you can mark the desired environment with the line command LO to load it into the debug buffer, or with the line command DE to delete it.

Save Debug Environment

To save a debug environment

- On the Debug Environment Maintenance menu, enter Function Code **S** and the name of an environment.
Or enter the direct command `SAVE ENVIRONMENT name`.
The specified environment is reset (see below) and saved to the file location specified with the `PROFILE` command as mentioned earlier.

If you do not specify a name, the environment is saved with the name Noname.

If a debug environment with the specified name already exists, you are prompted for confirmation to overwrite the old environment.

Reset Debug Environment

The debug environment should be reset before each test run. Resetting the environments leads to the following results:

- The current states of all debug entries are set to their initial states,
- All event counts are set to zero,
- The call statistics in the debug buffer are cleared as described in the section Call Statistics Maintenance.

To reset a debug environment

- On the Debug Environment Maintenance menu, enter Function Code **R** and the name of an environment.
Or enter the direct command `RESET ENVIRONMENT name`.
The specified debug environment is reset.

If you do not specify an environment name, the current debug environment is reset.

Delete Debug Environment

To delete a debug environment

1. On the Debug Environment Maintenance menu, enter Function Code **D** and the name of the environment.
Or enter the direct command `DELETE ENVIRONMENT name`.
The confirmation window appears.
2. In the confirmation window, enter Y (Yes) to confirm the deletion.
The debug specified environment is deleted.

If you do not specify an environment name, the current debug environment is deleted.

Spy Maintenance

The Spy Maintenance functions are used to activate, deactivate, list or delete **all** debug entries (spies) that is, breakpoints **and** watchpoints. Besides, Spy Maintenance is an alternative method of accessing the breakpoint or watchpoint maintenance screens.

▶ To invoke the Spy Maintenance menu

- On the Debug Main Menu, enter Function Code **S**.
Or enter the direct command **SM**.
The Spy Maintenance menu appears.

The Spy Maintenance menu provides the following functions:

- Set Test Mode ON/OFF
(see the relevant section)
 - Activate Spy
 - Deactivate Spy
 - Delete Spy
 - Display Spy
 - Modify Spy
-

Activate Spy

▶ To set the current state of specified spies to "active"

- On the Spy Maintenance menu, enter Function Code **A** and a spy number **or** a spy name.
Or enter the direct command `ACTIVATE SPY name or number`
(see also the syntax of `ACTIVATE` in Command Summary and Syntax).

If you do not specify an object name or a line number, **all** spies (breakpoints and watchpoints) are activated.

Deactivate Spy

▶ To set the current state of specified spies to "inactive"

- On the Spy Maintenance menu, enter Function Code **B** and a spy number **or** a spy name.
Or enter the direct command `DEACTIVATE SPY name or number`
(see also the syntax of `DEACTIVATE` in Command Summary and Syntax).

If you do not specify an object name or a line number, **all** spies (breakpoints and watchpoints) are deactivated.

Delete Spy

▶ To delete specified spies

- On the Spy Maintenance menu, enter Function Code **C** and a spy number **or** a spy name.
Or enter the direct command `DELETE SPY name or number`
(see also the syntax of `DELETE` in Command Summary and Syntax).

If you do not specify an object name or a line number, **all** spies (breakpoints and watchpoints) are deleted.

Display Spy

To display specified spies

- On the Spy Maintenance menu, enter Function Code **D** and a spy number **or** a spy name.
Or enter the direct command `DISPLAY SPY name or number`
(see also the syntax of `DISPLAY` in Command Summary and Syntax).

If the specified spy is unique, the Display Breakpoint or Display Watchpoint screen (see the relevant sections) appears respectively and all specifications of this breakpoint or watchpoint are displayed.

If the specified spy is not unique, a list of the spies concerned is displayed. On the list, you can activate, deactivate, display, modify or delete a spy by marking it with the line command `AC`, `DA`, `DI`, `MO` or `DE` respectively.

If you do not specify an object name or a line number, **all** spies (breakpoints and watchpoints) are displayed.

Modify Spy

To modify specified spies

- On the Spy Maintenance menu, enter Function Code **M** and a spy number **or** a spy name.
Or enter the direct command `MODIFY SPY name or number`
(see also the syntax of `MODIFY` in Command Summary and Syntax).

If the specified spy is unique, the Modify Breakpoint or Modify Watchpoint screen appears respectively and the breakpoint or watchpoint specifications can be modified.

If the specified spy is not unique, a list of the spies concerned is displayed. On the list, you can activate, deactivate, display, modify or delete a spy by marking it with the line command `AC`, `DA`, `DI`, `MO` or `DE` respectively.

If you do not specify an object name or a line number, **all** spies (breakpoints and watchpoints) are displayed for selection and modification.

Breakpoint Maintenance

A breakpoint causes the execution of a Natural object to be interrupted at a predefined statement line. This section describes how and when to set breakpoints. Note that the maintenance functions described here may also be invoked from an object source by using the function List Object Source.

Below is information on:

- Conditions of Use
 - Activate Breakpoint
 - Deactivate Breakpoint
 - Delete Breakpoint
 - Display Breakpoint
 - Modify Breakpoint
 - Set Breakpoint
-

Conditions of Use

A breakpoint is defined by specifying the name of the Natural object to be processed and the line number of the object's source code where the breakpoint is to be executed.

Once a breakpoint has been specified, it remains set for the entire Natural session, unless you delete it.

A breakpoint refers to a specific line number in a source code. A subsequent change of the source code itself may therefore lead to the breakpoint no longer applying to the desired statement, and thus the program not being interrupted at the desired position. To circumvent this problem with program loops, labels can be set within these loops. Breakpoints set for these labels are adjusted to the correct line number if statement lines are inserted or deleted.

The unique identifier for a breakpoint is the Spy Number as assigned by the Natural Debugger.

Breakpoints cannot be set in copycode, on comment lines, on any statement line other than the first one (if a single statement occupies more than one program line), and on lines that contain one of the following statements only:

- AT BREAK OF
- AT END OF DATA
- AT END OF PAGE
- AT START OF DATA
- AT TOP OF PAGE
- BEFORE BREAK
- DECIDE
(on lines with WHEN and VALUE clauses, however, breakpoints can be set)
- DEFINE SUBROUTINE
- DEFINE WINDOW
- FORMAT
- IF NO RECORDS FOUND
- ON ERROR
- OPTIONS

Whether it is possible or not to set breakpoints for lines compiled with the Natural Optimizer Compiler depends on the NODBG option of the OPTIONS statements as described in "Switching on the Optimizer Compiler" in the Natural Optimizer Compiler documentation.

Activate Breakpoint

▶ **To set the current state of specified breakpoints to "active"**

- On the the Breakpoint Maintenance menu, enter Function Code **A**, an object name and/or a line number.
Or enter the direct command `ACTIVATE BREAKPOINT object and/or line`
(see also the syntax of `ACTIVATE` in Command Summary and Syntax).

If you do not specify an object name or a line number, **all** breakpoints are activated.

Deactivate Breakpoint

▶ **To set the current state of specified breakpoints to "inactive"**

- On the the Breakpoint Maintenance menu, enter Function Code **B**, an object name and/or a line number.
Or enter the direct command `DEACTIVATE BREAKPOINT object and/or line`
(see also the syntax of `DEACTIVATE` in Command Summary and Syntax).

If you do not specify an object name or a line number, **all** breakpoints are deactivated.

Delete Breakpoint

▶ **To delete specified breakpoints**

- On the the Breakpoint Maintenance menu, enter Function Code **C**, an object name and/or a line number.
Or enter the direct command `DELETE BREAKPOINT object and/or line`
(see also the syntax of `DELETE` in Command Summary and Syntax).

If you do not specify an object name or a line number, **all** breakpoints are deleted.

Display Breakpoint

▶ **To display specified breakpoints**

- On the Breakpoint Maintenance menu, enter Function Code **D**, an object name and/or a line number.
Or enter the direct command `DISPLAY BREAKPOINT object and/or line`
(see also the syntax of `DISPLAY` in Command Summary and Syntax).

If a unique breakpoint has been specified, the Display Breakpoint screen appears and all specifications of this breakpoint are displayed. The Display Breakpoint screen is identical to the Modify Breakpoint screen. For an explanation of the fields, see Display/Modify Breakpoint Screen below.

If no unique breakpoint has been specified, a list displays all breakpoints set for the current environment. On the list, you can activate, deactivate, display, modify or delete a breakpoint by marking it with the line command AC, DA, DI, MO or DE respectively.

If you do not specify an object name or a line number, **all** breakpoints are displayed.

Modify Breakpoint

▶ **To modify specified breakpoints**

1. On the Breakpoint Maintenance menu, enter Function Code **M**, an object name and/or a line number.
Or enter the direct command **MODIFY BREAKPOINT** *object* and/or *line*
(see also the syntax of **MODIFY** in Command Summary and Syntax).
If a unique breakpoint has been specified, the Modify Breakpoint screen appears and the breakpoint specifications can be modified. The Modify Breakpoint screen is identical to the Display Breakpoint screen.
For an explanation of the fields, see Display/Modify Breakpoint Screen below.
If no unique breakpoint has been specified, a list displays all breakpoints set for the current environment.
On the list, you can activate, deactivate, display, modify or delete a breakpoint by marking it with the line command **AC**, **DA**, **DI**, **MO** or **DE** respectively.
2. On the Modify Breakpoint screen, choose **PF3/Exit** or **PF5/Save** to save any modification.
If you choose **PF12/Canc**, the breakpoint remains unchanged.

If you do not specify an object name or a line number, **all** breakpoints are displayed for selection and modification: see Display Breakpoint above.

Below is information on:

- Display/Modify Breakpoint Screen

Display/Modify Breakpoint Screen

The Modify Breakpoint screen provides the following fields:

Field	Explanation
Spy Number	A unique number assigned by the Natural Debugger when setting the breakpoint.
Initial State	Specifies the current state of the breakpoint: active or inactive.
Breakpoint Name	Valid input: range from 1 to 12 characters. The default name for breakpoints consists of the object name and the line number.
Object Name	The name of an object available in the current library or one of its steplibs. The default name is the name of the default object (see the section Start the Natural Debugger) if defined.
Line Number	The line number of a statement in the object source code. See also Conditions of Use above. You can also specify BEG, END or ALL as line numbers: BEG The breakpoint is to executed at the first statement executed. END The breakpoint is to be executed at the last statement executed. ALL The breakpoint is to be executed for each program line of the object specified.
Label	Refers to a label set earlier in the source code of an object for statements that define processing loops: see also Conditions of Use above. Valid input: range from 1 to 32 characters.
Skips before Execution	Determines that the breakpoint is not to be executed until the corresponding statement line has been executed a certain number of times. Valid input: range from 0 (default) to 32767.
Number of Executions	Any value greater than zero (0) determines the maximum number of breakpoint executions. Valid input: range from 0 (default) to 32767.
Commands	Up to six debug commands. Enter one command per line. For a summary of all available commands, see Command Summary and Syntax. Attention: If you delete the command BREAK when modifying a breakpoint and you do not enter any command that issues a dialog, there is no way for the Natural Debugger to receive control during program interruption.

Set Breakpoint

To add a breakpoint to a session

- On the Breakpoint Maintenance menu, enter Function Code **S**, an object name and/or a line number.
Or enter the direct command SET BREAKPOINT *object* and/or *line*
(see also the syntax of SET in Command Summary and Syntax).

If object name and line number are specified correctly, the breakpoint is set immediately and a corresponding confirmation message is displayed on the screen. The breakpoint receives the default command (BREAK), its initial and current state are set to "active" and no execution restrictions are specified. Note that if you delete the

command **BREAK** when setting a breakpoint and you do not enter any command that issues a dialog, there is no way for the Natural Debugger to receive control during program interruption.

If you specify not an object name but a valid line number, the name of the default object (see the section **Start the Natural Debugger**) is assumed and the breakpoint is also set immediately. If there is no default object defined, a selection window appears that displays all objects available in the current library.

Watchpoint Maintenance

A watchpoint causes the execution of a Natural object to be interrupted whenever the value of a variable changes. In addition, you can make the interruption dependent on a condition related to a specific variable value as described under Watchpoint Operators (Set Watchpoint) below.

The use of watchpoints allows you to detect unintended alterations of variables caused by objects that contain errors.

A variable is considered to have changed either when its current value differs from the value recorded when the watchpoint was last triggered or when it differs from the initial value.

A watchpoint is defined by specifying the name of the Natural object or GDA (global data area) to be processed and the local or global variable it is to refer to.

The unique identifier for a breakpoint is the Spy Number as assigned by the Natural Debugger.

Once a watchpoint has been specified, it remains set for the entire Natural session, unless you delete it.

Below is information on:

- Invoke Watchpoint Maintenance
 - Activate Watchpoint
 - Deactivate Watchpoint
 - Delete Watchpoint
 - Display Watchpoint
 - Modify Watchpoint
 - Set Watchpoint
-

Invoke Watchpoint Maintenance

 **To invoke the Watchpoint Maintenance menu**

- On the Debug Main Menu, enter Function Code **W**.
Or enter the direct command **WM**.

The Watchpoint Maintenance menu provides the following functions:

- Set Test Mode ON/OFF
(see the relevant section)
- Activate Watchpoint
- Deactivate Watchpoint
- Delete Watchpoint
- Display Watchpoint
- Modify Watchpoint
- Set Watchpoint

Activate Watchpoint

 **To set the current state of specified watchpoints to "active"**

- On the the Watchpoint Maintenance menu, enter Function Code **A**, an object name and a variable name.
Or enter the direct command `ACTIVATE WATCHPOINT object and variable`
(see also the syntax of `ACTIVATE` in Command Summary and Syntax).

If you do not specify an object or a variable (or leave the default asterisk in the field Variable), **all** watchpoints are activated.

Deactivate Watchpoint

▶ **To set the current state of specified watchpoints to "inactive"**

- On the the Watchpoint Maintenance menu, enter Function Code **B**, an object name and a variable name.
Or enter the direct command `DEACTIVATE WATCHPOINT object and variable`
(see also the syntax of `DEACTIVATE` in Command Summary and Syntax).

If you do not specify an object name or a variable (or leave the default asterisk in the field Variable), **all** watchpoints are deactivated.

Delete Watchpoint

▶ **To delete specified watchpoints**

- On the Watchpoint Maintenance menu, enter Function Code **C**, an object name and a variable name.
Or enter the direct command `DELETE WATCHPOINT object and variable`
(see also the syntax of `DELETE` in Command Summary and Syntax).

If you do not specify an object name or a variable (or leave the default asterisk in the field Variable), **all** watchpoints are deleted.

Display Watchpoint

▶ **To display specified watchpoints**

- On the Watchpoint Maintenance menu, enter Function Code **D**, an object name and/or a variable name.
Or enter the direct command `DISPLAY WATCHPOINT object and/or variable`
(see also the syntax of `DISPLAY` in Command Summary and Syntax).

If a unique watchpoint has been specified, the Display Watchpoint screen appears and all specifications of this watchpoint are displayed. The Display Watchpoint screen is identical to the Modify Watchpoint screen. For an explanation of the fields, see Display/Modify Watchpoint Screen below.

If no unique watchpoint has been specified, a list displays all watchpoints set for the current environment.

On the list, you can activate, deactivate, display, modify or delete a watchpoint by marking it with the line command AC, DA, DI, MO or DE respectively.

If you want to know whether, or what kind of, a condition for the watchpoint to be activated has been specified, you can display the corresponding value and watchpoint operator (see below) by choosing either PF10 (if you want to display the value in alphanumeric format) or PF11 (if you want to display the value in hexadecimal format). PF22 takes you back to the Display Watchpoint screen.

If you do not specify an object name or a variable (or leave the default asterisk in the field Variable), **all** watchpoints are displayed.

Modify Watchpoint

▶ To modify specified watchpoints

- On the Watchpoint Maintenance menu, enter Function Code **M**, an object name and a variable name. Or enter the direct command `MODIFY WATCHPOINT object and a variable name` (see also the syntax of `MODIFY` in Command Summary and Syntax).

If a unique watchpoint has been specified, the Modify Watchpoint screen (see below) appears and the watchpoint specifications can be modified. The Modify Watchpoint screen is identical to the Display Watchpoint screen. For an explanation of the fields, see Display/Modify Watchpoint Screen below.

If no unique watchpoint has been specified, a list displays all watchpoints set for the current environment. On the list, you can activate, deactivate, display, modify or delete a watchpoint by marking it with the line command AC, DA, DI, MO or DE respectively.

If you want to modify the condition for the watchpoint to be activated, you can modify the corresponding value and watchpoint operator (see below) by choosing either PF10 (if you want to modify the value in alphanumeric format) or PF11 (if you want to modify the value in hexadecimal format). PF22 takes you back to the Modify Watchpoint screen (see Display/Modify Watchpoint Screen below).

- On the Modify Watchpoint screen (see Display/Modify Watchpoint Screen below), choose PF3/Exit or PF5/Save to save any modification. If you choose PF12/Canc, the watchpoint remains unchanged.

If you do not specify an object name or a variable (or leave the default asterisk in the field Variable), **all** watchpoints are displayed for selection and modification.

Below is information on:

- Display/Modify Watchpoint Screen

Display/Modify Watchpoint Screen

Field	Explanation
Spy Number	A unique number assigned by the Natural Debugger when setting the watchpoint.
Initial State	Specifies the initial state of the watchpoint: active or inactive. Default: active
Watchpoint Name	Valid input: range from 1 to 12 characters. The default name for watchpoints is the name of the variable concerned. Names that exceed the field size will be truncated after 12 characters.
Object Name	The name of an object available in the current library or one of its steplibs.
Variable Name	The name of a user-defined, global or system variable. If the variable is part of a structure, it may be prefixed by the structure name. For an array, an index description has to be specified (watchpoints can be defined for single elements only). Use Asterisk (*) notation for a selection list of variables. See also Variable Maintenance for further details.
Skips before Execution	Determines that the watchpoint is not to be executed until the condition set for the watchpoint has been fulfilled (see also Watchpoint Operators below). Valid input: range from 0 (default) to 32767.
Number of Executions	Any value greater than zero (0) determines the maximum number of watchpoint executions. Valid input: range from 0 (default) to 32767.
Commands	Up to six debug commands. Enter one command per line. For a summary of all available commands, see Command Summary and Syntax. Attention: If you delete the command BREAK and you do not enter any command that issues a dialog, there is no way for the Natural Debugger to receive control during program interruption.

Set Watchpoint

To add a watchpoint to a session

- On the Watchpoint Maintenance menu, enter Function Code **S**, an object name and a variable name.
Or enter the direct command SET WATCHPOINT *object* and *variable*
(see also the syntax of SET in Command Summary and Syntax).

If object name and variable names are specified correctly, the watchpoint is set immediately and a corresponding confirmation message is displayed on the screen. The watchpoint receives the default command (BREAK), its initial and current state are set to "active" and no execution restrictions are specified. Note that if you delete the default command BREAK when setting a watchpoint and you do not enter any command that issues a dialog, there is no way for the Natural Debugger to receive control during program interruption.

If you specify not an object name but a valid variable name, the name of the default object (see the section Start the Natural Debugger) is assumed and the watchpoint is also set immediately. If there is no default object defined, a selection window appears that displays all objects available in the current library.

Below is information on:

- Watchpoint Operators

Watchpoint Operators

 **To specify a condition for the watchpoint to be activated**

- On the Set Watchpoint or Modify Watchpoint screen, choose PF10 if you want to specify the value in alphanumeric format,
Or, on the Set Watchpoint or Modify Watchpoint screen, choose PF11 if you want to specify the value in hexadecimal format.
- In the left input field, enter any of the watchpoint operators listed below.
In the right fields that, enter the value to be compared with the variable.
- Choose PF22 to return to the Set Watchpoint screen.

Watchpoint operators are:

Operator	Explanation
MOD	Modification. Indicates that the watchpoint is activated each time a modification of the variable occurs. This is the default setting.
EQ	Equal. Indicates that the watchpoint is activated only when the variable has been modified and when the current value of the variable is equal to the specified value.
NE	Not Equal. Indicates that the watchpoint is activated only when the variable has been modified and when the current value of the variable is not equal to the specified value.
GT	Greater Than. Indicates that the watchpoint is activated only when the variable has been modified and when the current value of the variable is greater than the specified value.
GE	Greater or Equal. Indicates that the watchpoint is activated only when the variable has been modified and when the current value of the variable is greater than or equal to the specified value.
LT	Less Than. Indicates that the watchpoint is activated only when the variable has been modified and when the current value of the variable is less than the specified value.
LE	Less or Equal. Indicates that the watchpoint is activated only when the variable has been modified and when the current value of the variable is less than or equal to the specified value.

Call Statistics

The Call Statistics function can be used to obtain statistical information on which programming objects were invoked during the execution of an application, and information on how often an object was invoked. Call statistics are deleted after resetting the debug environment.

▶ To invoke the Call Statistics Maintenance function

- On the Debug Main Menu enter Function Code **C**.
Or enter the direct command **CS**.
The Call Statistics Maintenance screen is displayed which provides the following functions:
 - Set Test Mode ON/OFF
(see the relevant section)
 - Set Call Statistics ON/OFF
 - Display All Objects
 - Display Called Objects
 - Display Non-Called Objects
 - Print All Objects
 - Print Called Objects
 - Print Non-Called Objects

The print functions above are described under Print Objects below.

Set Call Statistics ON/OFF

When starting a program with Call Statistics ON or when activating the call statistics while running a program, all calls made to a specific object are counted and the resulting statistics can afterwards be displayed or printed.

▶ To activate or deactivate Call Statistics

- On the Call Statistics Maintenance menu enter Function Code **C** to activate and deactivate.
Or enter the direct command **SET CALL ON** or **SET CALL OFF**.

Display All Objects

The Display Call Statistics screen provides an overview of the call frequency of the objects in your current library.

▶ To display all objects of a library

- On the Display Call Statistics Maintenance screen, enter Function Code **1** and a library name.
Or enter the direct command **DISPLAY OBJECT *library***
(see also the syntax of **DISPLAY** in Command Summary and Syntax).
If you do not specify a library name, the library where you are currently logged on is assumed by default.

The Display Call Statistics screen lists all objects in your current library and indicates their call frequency in the Calls column on the right-hand side. For each call statement, such as **FETCH** or **CALLNAT**, an entry with the name of the object and a counter variable is written into the debug buffer. The counter is then increased for each call of the corresponding object.

For an example screen, see Display Called Objects below.

Display Called Objects

The Display Called Objects screen corresponds to the Display Call Statistics screen, but only the objects that have been invoked are displayed.

▶ To display called objects of a library

- On the Call Statistics Maintenance menu, enter Function Code **2** and a library name.
Or enter the direct command `DISPLAY CALL library`
(see also the syntax of `DISPLAY` in Command Summary and Syntax).
The Display Called Objects screen appears:

16:06:53		***** NATURAL TEST UTILITIES *****							2002-02-15	
Test mode ON		- Display Called Objects -							Object	
Object	Library	Type	DBID	FNR	S/C	Ver	Cat	Date	Time	All Calls
*_____	SAG_____									
MAINPGM	SAG	Program	10	32	S/C	3.1		2002-02-15	11:51	1
SUBPGM	SAG	Subprogram	10	32	S/C	3.1		2002-02-15	11:50	3
EMP-PGM	SAG	Program	10	32	S/C	3.1		2002-01-22	11:49	2
EMPLIND	SAG	Program	10	32	S/C	3.1		2001-08-13	11:18	1

If you do not specify a library name, the library where you are currently logged on is assumed by default.

Display Non-Called Objects

The Display Non-Called Objects screen corresponds to the Display Call Statistics screen, but only the objects that have **not** been invoked are displayed.

▶ To display non-called objects

- On the Call Statistics Maintenance menu, enter Function Code **3** and a library name.
Or enter the direct command `DISPLAY NOCALL library`
(see also the syntax of `DISPLAY` in Command Summary and Syntax).

If you do not specify a library name, the library where you are currently logged on is assumed by default.

For an example screen, see Display Called Objects above.

Print Objects

With the print functions, you can directly route a generated list of call statistics to a printer.

If you do not specify a library name, the library where you are currently logged on is assumed by default.

As listed under Print Options below, to invoke one of the print functions, you can either enter a function code on the Call Statistics Maintenance menu, or enter a direct command:

Print Options

Print Function	Function Code	Direct Command
All Objects	4	PRINT OBJECT <i>library</i> .
Called Objects	5	PRINT CALL <i>library</i> .
Non-Called Objects	6	PRINT NOCALL <i>library</i> .

See also the syntax of PRINT in Command Summary and Syntax.

Statement Execution Statistics

The Statement Execution Statistics function is used to obtain statistical information on which statement lines of invoked programming objects were executed. The function also provides information on how often an object was invoked and how often a statement line was executed.

Statement execution statistics can be used for

- detecting dead (never gets executed) programming code in an application.
- estimating the coverage of an application test (how many statement lines have not been executed at least once for testing).
- locating frequently executed code segments that could have an impact on the application's performance.

This section covers the following topics:

- Set Statement Execution Statistics ON/OFF/COUNT
 - Invoke Statement Execution Statistics
 - Delete Statement Execution Statistics
 - Display Statement Execution Statistics
 - Print Statements
-

Set Statement Execution Statistics ON/OFF/COUNT

With this function, you activate statistics about executed statement lines of programming objects.

Below is information on:

- Setup Options
- Activate/deactivate Statistics

Setup Options

When starting a program with Statement Execution Statistics ON or COUNT, all statement lines executed within a specific object are listed in a statistical report.

With the option ON, the Debugging utility only retains whether a specific statement line was executed or not; with the COUNT option, it counts how often a statement line was executed. You can specify a library and an object name to restrict statement execution statistics to the desired programming objects. The default is to collect statistics for all objects of the current library. Asterisk (*) notation is possible.

If you switch Statement Execution Statistics from ON to COUNT or vice versa, existing statistics are not affected, that is, their status of ON or COUNT remains.

The statistical data collected is stored in the debug buffer. The amount of storage that is required to store statistical information for a programming object is approximately

(number of source lines) / 8 + 100 bytes with Statement Execution Statistics ON and
(number of source lines) * 4 + 100 bytes with Statement Execution Statistics COUNT.

If you modify a programming object by inserting or deleting lines and you do not renumber the object lines before you STOW it, the amount of storage required for the object's statistics may increase. To avoid this, set Auto Renumber to Y (Yes) in the editor profile or use the command CATALL with the function Renumber source code lines option enabled (this is the default).

You can use the direct command PROFILE to limit the size of the debug buffer. With Statement Execution

Statistics set to option COUNT, no statement execution statistics are collected for objects with more than 8000 statement lines.

Statement Execution Statistics are part of the debug environment; therefore, they are affected by the direct commands SAVE ENVIRONMENT and LOAD ENVIRONMENT (see also the section Debug Environment Maintenance).

Activate/deactivate Statistics

Below are instructions on how to activate or deactivate the function Statement Execution Statistics Maintenance. You can specify a library and/or an object name to restrict statement execution statistics to the desired programming objects. The default is to collect statistics for all objects of the current library. Asterisk (*) notation is possible.

▶ To activate Statement Execution Statistics

- On the Statement Execution Statistics Maintenance screen, enter Function Code **S**, the name of a library and/or the name of an object. In the State field, change the value to ON.
Or enter the direct command SET XSTATISTICS ON or COUNT *library (object)*
(see also the syntax of SET in Command Summary and Syntax).

If you do not specify a library and/or an object, the statistics data about all objects in your current library are activated.

▶ To deactivate Statement Statement Execution Statistics

- On the Statement Execution Statistics Maintenance screen, enter Function Code **S**, the name of a library and/or the name of an object. In the State field, change the value to OFF.
Or enter the direct command SET XSTATISTICS OFF *library (object)*
(see also the syntax of SET in Command Summary and Syntax).

If you do not specify a libber and/or an object, the statistics data about all objects in your current library are deactivated.

Invoke Statement Execution Statistics

▶ To invoke the Statement Execution Statistics Maintenance function

- On the Debug Main Menu, Function Code **X**.
Or enter the direct command XS.
The Statement Execution Statistics Maintenance screen is displayed which provides the following functions:
 - Set Test Mode ON/OFF
(see the relevant section)
 - Set Statement Execution Statistics ON/OFF/COUNT
 - Delete Statement Execution Statistics
 - Display Statement Execution Statistics
 - Print Statement Execution Statistics
 - Print All Statements
 - Print Executed Statement
 - Print Non-Executed Statements

The print functions are described under Print Statements below.

Delete Statement Execution Statistics

▶ To delete Statement Execution Statistics

- On the Statement Execution Statistics Maintenance menu, enter Function Code **C** and the name of a library and/or the name of an object.
Or enter the direct command `DELETE XSTATISTICS library (object)`
(see also the syntax of `DELETE` in Command Summary and Syntax).

If you do not specify a library and/or an object, the statistics data about all objects in your current library are deleted.

Display Statement Execution Statistics

The List Statement Execution Statistics screen displays a list of the specified statement execution statistics.

▶ To invoke the List Statement Execution Statistics screen

- On the Statement Execution Statistics Maintenance menu, enter Function Code **D**.
Or enter direct command `DISPLAY XSTATISTICS`.
The List Statement Execution Statistics screen is displayed:

16:02:01		***** NATURAL TEST UTILITIES *****						2002-02-15		
Test Mode ON		- List Statement Execution Statistics -						Object		
Co	Object	Library	Type	DBID	FNR	Obj.Called	Exec	Exec	%	Total No.
	*	*				n Times	able	uted		Executions
___	PGM01	SAG	Program	10	32	4	20	17	85	95
___	MAP01	SAG	Map	10	32	6	2	2	100	12
___	SPGM02	SAG	Subprogram	10	32	2	6	2	33	4
___	SAGTEST1	SAG	Program	10	32	2	20	10	50	17
___	DEBPGM	SAG	Program	10	32	1	6	6	100	34

For each object, the following information is displayed:

- the call frequency;
- the number of executable statement lines (a statement line is executable if a breakpoint could be set on it; see the description of the command `SET BREAKPOINT` in the section Breakpoint Maintenance for more information);
- the number of executed statement lines;
- the percentage of executed statement lines as related to the total number of executable statement lines;
- the total number of executed statement lines.

A list entry is highlighted if data is missing or possibly inconsistent.

- On the statistics list, you can mark statement execution statistics with a line command for further processing:

Command	Explanation
DE	Deletes statement execution statistics as described above.
DS	Displays all statement lines.
DX	Displays executed statement lines only.
DN	Displays non-executed statement lines only.
I	Displays information on the cataloged object and errors.
PS	Prints all statement lines.
PX	Prints executed statement lines only.
PN	Prints non-executed statement lines only.

The print functions are also described under Print Statements below.

Below is information on:

- Display All Statement Lines
- Display Executed Statement Lines
- Display Non-Executed Statement Lines

Display All Statement Lines

The Display Statement Lines screen shows the object source and indicates whether or not a statement line has been executed.

To invoke the Display Statement Lines screen

- On the List Statement Execution Statistics screen, mark an entry with the line command **DS**. Or enter the direct command `DISPLAY STATEMENT library (object)` (see also the syntax of `DISPLAY` in Command Summary and Syntax).

The Display Statement Lines screen appears. If Statement Execution Statistics has been set to `COUNT`, the execution frequency of the statement line is displayed as shown in the example screen below:

```

16:04:01          *****NATURAL TEST UTILITIES *****          2002-02-15
Test Mode ON          - Display Statement Lines -          Object SAGTEST

Line Source                                     Count
0200  RD1. READ EMPLOYEES-VIEW BY NAME                2
0210          STARTING FROM #NAME-START THRU #NAME-END
0220 *
0230  IF LEAVE-DUE>= 20                                1
0240          PERFORM MARK-SPECIAL-EMPLOYEES          not executed
0250  ELSE                                             not executed
0260          RESET #MARK                              1
0270  END-IF
0280 *
0290  RESET #MAKE #MODEL                              1
0300  CALLNAT 'SPGM02' PERSONNEL-ID #MAKE #MODEL      1
0310 *
0320  WRITE TITLE / '*** PERSONS WITH 20 OR MORE DAYS LEAVE DU
0330  / '*** ARE MARKED WITH AN ASTERISK          ***' //
0340  DISPLAY '//N A M E' NAME                        2
    
```

If no unique object has been specified, the List Statement Execution Statistics screen is displayed.

Display Executed Statement Lines

The Display Executed Statement Lines screen corresponds to the Display Statement Lines screen, but only the statement lines that have been executed are displayed.

To invoke the Display Executed Statement Lines screen

- On the List Statement Execution Statistics screen, mark an entry with the line command **DX**.
Or enter the direct command `DISPLAY EXEC library (object)`
(see also the syntax of DISPLAY in Command Summary and Syntax).

If no unique object has been specified, the List Statement Execution Statistics screen is displayed.

Display Non-Executed Statement Lines

The Non-Executed Statement Lines screen corresponds to the Display Statement Lines screen, but only the statement lines that have not been executed are displayed.

To invoke the Display Non-Executed Statement Lines screen

- On the List Statement Execution Statistics screen, mark an entry with the line command **DN**.
Or enter the direct command `DISPLAY NOEXEC library (object)`
(see also the syntax of DISPLAY in Command Summary and Syntax).

If no unique object has been specified, the List Statement Execution Statistics screen is displayed.

Print Statements

With the print functions, you can directly route a generated list of statement execution statistics to a printer.

If you do not specify a library name, the library where you are currently logged on is assumed by default.

As listed under Print Options below, to invoke one of the print functions, you can either enter a function code on the Statement Execution Statistics Maintenance menu, enter a line command on the Display Statement Lines screen, or enter a direct command:

Print Options

Print Function	Function Code	Line Command	Direct Command
Statement Execution Statistics	1		<code>PRINT XSTATISTICS <i>library (object)</i></code> .
All Statements	2	PS	<code>PRINT STATEMENT <i>library (object)</i></code> .
Print Executed Statements	3	PX	<code>PRINT EXEC <i>library (object)</i></code> .
Print Non-Executed Statements	4	PN	<code>PRINT NOEXEC <i>library (object)</i></code> .

See also the syntax of PRINT in Command Summary and Syntax.

Variable Maintenance

With the Variable Maintenance function, you can display and modify variables within the Natural Debugger if a program was interrupted.

For the interrupted object, the variable function displays user-defined variables, global variables and the database-related system variables *COUNTER, *ISN and *NUMBER, together with their formats, lengths and contents.

Below is information on:

- Display Variable
 - Modify Variable
-

Display Variable

▶ To display all user-defined and global variables and the database-related system variables

- On the Debug Main Menu or in the Debug Window, enter Function Code **V**.
Or enter the direct command `DISPLAY VARIABLE` or `MODIFY VARIABLE` without specifying a variable.
The Display Variables summary screen appears with a list of all variables defined for the interrupted program.
Long values may be displayed truncated on the screen. For arrays, only the contents of the first occurrence is displayed.
- To display a variable value in its entirety:
Select the desired variable by marking it with the line command **DI**.
The Display Variable screen for the individual variable appears with all relevant specifications.
- To display all occurrences of an array:
Select the desired variable by marking it with the line command **DI**.
 - Use PF7 (-) and PF8 (+) to page between the individual occurrences or
 - Enter the direct command `DIS VAR variable(*)` to display all occurrences.

On the Display Variables (summary) or the Display Variable (individual) screen, you can toggle between alphanumeric and hexadecimal representation of the variable contents by using PF10 (Alpha) and PF11 (Hex).

▶ To display a selection of variables

- Enter the direct command `DISPLAY VARIABLE variable,variable,...`
The Display Variables summary screen appears with a list of the variables specified.

▶ To display system variables (except database-related variables)

- Enter the direct command `SYSVARS`.
The System Variables screen appears with a list of all system variables.

For variables of the type Handle, the name of the class of the instance that the Handle refers to is displayed in alphanumeric representation. If the class name is not available, the Globally Unique ID (GUID) is displayed instead. If the class was defined within Natural, the class name or GUID is suffixed with "(NAT)".

The contents of properties of an instance of a class cannot be displayed within the Natural Debugger.

Modify Variable

Not applicable to system variables.

With this function, you can change the value of user-defined and global variables and the database-related system variables.

To modify the contents of a variable from the Modify Variable screen

- Invoke the Modify Variable screen by marking the variable with the line command **MO**.
Or, on the Display Variable screen, choose PF5.
- On the Modify Variable screen, in the field Contents, change the value of the variable.
The new contents must be valid for the format of the modified variable since the format of a variable cannot be modified within the Natural Debugger.
On the Modify Variable screen, you can toggle between alphanumeric and hexadecimal representation of the variable value using PF10 (Alpha) and PF11 (Hex).

To modify the contents of a variable via direct command

- Enter the direct command `MODIFY VARIABLE variable = new value`
(see also the syntax of **MODIFY** in Command Summary and Syntax).
A message appears that confirms modification of the variable value.

List Object Source

With the List Object Source function, you can display the source code of an object and maintain breakpoints. For you to be able to use List Object Source, the corresponding source must be in your current library or in one of its steplibs.

▶ To list the source code of an object

- On the Debug Main Menu, enter Function Code L and an object name.
Or enter the direct command `LIST object`
(see also the syntax of LIST in Command Summary and Syntax).
The object source is displayed with all current breakpoints listed in the Message column on the right-hand side of the screen.
Use PF8 or PF9 to scroll up or down one page.

If you execute a programming object, the Natural Debugger interrupts execution at each breakpoint or watchpoint you have set and the Debug Window appears (see the relevant section in Concepts of the Natural Debugger).

▶ To list the source code of an interrupted object

- From the Debug Window, choose Function Code L for List Break.
Or, if relevant on other debug screens, choose PF9 (Li Br) or enter the direct command `LIST BREAK`.
The source code of the object is displayed at the position where a break (breakpoint or watchpoint) occurred. The name of the breakpoint or watchpoint is displayed in the Message column on the right-hand side of the screen. The corresponding source code line is highlighted.

Below is information on:

- Maintain Breakpoints
-

Maintain Breakpoints

The List Object Source function, may be used to invoke or directly execute breakpoint maintenance functions from within an object source. For instructions on how to set breakpoints and general information on breakpoints, see Conditions of Use in Breakpoint Maintenance.

▶ To invoke a breakpoint maintenance function from an object source

- On the Debug Main Menu, enter Function Code L and an object name.
Or enter the direct command `LIST object`
(see also the syntax of LIST in Command Summary and Syntax).
The source code of the specified object is displayed.
The names of breakpoints already set are displayed in the Message column on the right-hand side of the screen.
 - To scroll the listing:
 - In the command line, enter a plus (+) or a minus (-) sign
or enter the direct commands TOP, BOTTOM, LEFT and RIGHT.
- In the object source, mark the line(s) desired with any of the commands listed below:

AC	Activates breakpoints.
DA	Deactivates breakpoints.
DE	Deletes breakpoints.
DI	Displays breakpoints.
MO	Goes to the Modify Breakpoint maintenance screen.
SE	Sets breakpoints.
SM	Goes to the Set Breakpoint maintenance screen.

- Upon successful command execution, a corresponding message is displayed in the Message column on the right-hand side of the screen.

Execution Control Commands

Listed below are commands the Natural Debugger provides for controlling the program flow during a debugging session. For a summary of all commands available with the Natural Debugger, refer to Command Summary and Syntax.

The commands listed below only apply when the Natural Debugger interrupts program execution.

- ESCAPE BOTTOM
 - ESCAPE ROUTINE
 - EXIT
 - GO
 - NEXT
 - RUN
 - STEP
 - STEP SKIPSUBLEVEL
 - STOP
-

ESCAPE BOTTOM

This command can only be used when an object has been interrupted within a processing loop.

When you enter this command, the interrupted object will be continued with the first statement following the processing loop.

ESCAPE ROUTINE

When you enter this command, processing of the interrupted object will be stopped and processing will continue with the object from which the interrupted object was invoked; it will continue with the statement following the corresponding CALLNAT, PERFORM or FETCH RETURN statement.

If you apply the command ESCAPE ROUTINE to a main program, Natural ends the program and returns to the command mode.

EXIT

If you are displaying the Debug Main Menu and invoke the EXIT function, choose PF3 (Exit) or enter the execution control command EXIT, the debugging utility returns either to the calling program (that is, to the interrupted Natural object which is then continued) or to the NEXT line if the debugging utility has been invoked with the direct command TEST, or to the corresponding input field if it has been invoked by the terminal command %<TEST. However, if a breakpoint or watchpoint is currently active, the next command of this breakpoint or watchpoint is executed.

If you are not in the Debug Main Menu and enter the direct command EXIT or choose PF3 (Exit), you leave the current function and return to the previous step of your debugging session.

GO

When you enter the direct command GO (or choose PF14), the debugging utility returns control to the execution of the interrupted program. If a breakpoint or watchpoint was active at the time the program was interrupted, the remaining commands of this break or watchpoint are **not** executed.

NEXT

When you enter the direct command NEXT (or choose PF13), the next command specified for a breakpoint or watchpoint is executed. If no further command has been specified, program execution continues.

RUN

When you enter the direct command RUN, test mode is switched off and program execution continues, without investigating any further breakpoints and watchpoint.

STEP

When you enter the direct command STEP (or choose PF2), an interrupted Natural object is continued for *n* statement lines. The default value for *n* is 1.

STEP SKIPSUBLEVEL

When you enter the direct command STEP SKIPSUBLEVEL (or choose PF17) upon a statement which invokes another object (for example, CALLNAT), processing is continued with the next statement line in the interrupted Natural object (instead of the first executed statement in the invoked object).

With the command, you can specify a level number *n*. *n* may be the level of the interrupted object (this is the default) or a superior level. Step mode then continues with the next object at the specified level.

If this command is applied to a statement that does not invoke another object, the debugging utility reacts as if the command STEP had been entered.

Level information can be obtained with the command OBJCHAIN as described in the section Navigation and Information.

STOP

When you enter the direct command STOP, both the debugging utility and any interrupted Natural object are terminated; the NEXT line is displayed.

Navigation and Information Commands

Listed below are direct commands the Natural Debugger provides for navigating through the debugging areas, scrolling screen displays, obtaining various information on objects and variables, and specifying profiles. For a summary of all commands available with the Natural Debugger, refer to Command Summary and Syntax.

- BREAK
 - FLIP
 - LAST
 - OBJCHAIN
 - ON/OFF
 - PROFILE
 - SCAN
 - SCREEN
 - SET OBJECT
 - STACK
 - SYSVARS
 - TEST ON/OFF
-

BREAK

The command BREAK is the default command which is automatically set when creating a new debug entry. It displays the Debug Window as described in Concepts of the Natural Debugger.

When the command BREAK is deleted upon modification of the corresponding debug entry, no Debug Window appears. However, other specified commands are executed and the event count is increased.

FLIP

The command FLIP switches between the display of the two PF-key lines (PF1 to PF12 and PF13 to PF24).

LAST

The command LAST displays the command last entered. The last three commands are stored and can be recalled.

OBJCHAIN

The command OBJCHAIN can only be used when an object has been interrupted.

This command displays the objects on the current level and all superior levels, as well as the current GDA (Global Data Area), if applicable, and provides information on the interruption.

ON/OFF

When you enter the command ON/OFF in the Natural debug utility, test mode is switched on or off respectively. See also TEST ON/OFF below.

PROFILE

The command PROFILE displays the Edit Profile screen where you can modify the profile of the debugging utility.

Edit Profile Screen

The Edit Profile screen provides you with the following options:

Option	Explanation
Reset Debug Environment Automatically on Exit	Specifies an automatic reset of your current debug environment once you exit the debugging utility. The default is N (No).
File for Loading/Saving Debug Environments	Specifies to/from which system file debug environments are to be saved/loaded: FUSER (default), FNAT or SPAD (scratch-pad file).
Confirm EXIT/CANCEL Before Execution	Specifies a confirmation of an EXIT or CANCEL command before execution. The default is N (No).
Stack Unknown Commands	Specifies that any unknown debug command which is entered (for example, the name of a called program) is to be stacked. If so, once you enter an unknown debug command, you immediately exit the debugging utility and the command is executed. If this option has not been specified, an unknown debug command leads to a corresponding error message. The default is Y (Yes).
Output Device	Specifies an output device for the Call Statistics Maintenance function; the default value is HARDCOPY.
Maximum debug buffer size in KB	Specifies the maximum size (in kilobytes) of the debug buffer. The debug buffer is automatically enlarged as required, but only up to the specified maximum. Enter 0 to indicate no limit or enter a value from 4 - 16384 (must be a multiple of 4). If the limit would be exceeded, no further debug entries can be defined and no additional call or statement execution statistics entries are generated.

SCAN

Only applies to the List Object Source (see the relevant section) function.

This command searches for a string of characters within an object source:

SCAN searches for the value specified which may be delimited by blanks or by any characters that are neither letters nor numeric characters.

SCAN ABS results in an absolute scan of the source code for the specified value regardless of what other characters may surround the value.

See also the syntax diagrams in Command Summary and Syntax.

SCREEN

When you enter the command SCREEN upon interruption of a Natural object, the current screen output of the interrupted object is displayed. ENTER takes you back to debug mode.

SET OBJECT

The command SET OBJECT changes the name of the default object as described in the relevant section in Start the Natural Debugger. See also the syntax of SET in the section Command Summary and Syntax.

STACK

When you enter the command STACK, the contents of the entry at the top of the Natural stack is displayed. Up to 15 individual top entry elements can be displayed. Elements longer than 55 characters are truncated and marked with an asterisk (*).

Note:

An error message is displayed if any single element is longer than 249 characters.

SYSVARS

When you enter this command, the current values of system variables are displayed.

TEST ON/OFF

The command TEST ON/OFF switches test mode on or off respectively. In the Natural debug utility, you only need to enter ON/OFF as described above.

Command Summary and Syntax

This section provides a list of all commands available with the Natural Debugger. An underlined portion of a keyword represents an acceptable abbreviation. For an explanation of more complex command structures with user-defined operands, see Syntax Diagrams below.

Below is information on:

- All Debug Commands
- Syntax Diagrams

All Debug Commands

Command	Subcommand(s)	Explanation
-		Scrolls one page down in a list.
--		Scrolls to the beginning of a list.
<u>TOP</u>		
+		Scrolls one page down in a list.
++		Scrolls to the end of a list.
<u>BOTTOM</u>		
<u>ACTIVATE</u> (syntax below)	<u>BREAKPOINT</u> <u>BP</u>	Activates breakpoints as described in the relevant section in Breakpoint Maintenance.
	<u>SPY</u>	Activates breakpoints and watchpoints: see also Activate Spy in Spy Maintenance.
	<u>WATCHPOINT</u> <u>WP</u>	Activates watchpoints as described in the relevant section in Watchpoint Maintenance.
<u>BREAK</u>		Displays the Debug Window: see also <u>BREAK</u> in Navigation and Information Commands.
<u>CANCEL</u>		Cancels the current operation and/or exits screens without saving modifications.
<u>DEACTIVATE</u> <u>DA</u> (syntax below)	<u>BREAKPOINT</u> <u>BP</u>	Deactivates breakpoints as described in the relevant section in Breakpoint Maintenance.
	<u>SPY</u>	Deactivates breakpoints and watchpoints: see also Deactivate Spy in Spy Maintenance.
	<u>WATCHPOINT</u> <u>WP</u>	Deactivates watchpoints as described in the relevant section in Watchpoint Maintenance.

Command	Subcommand(s)	Explanation
<u>DELETE</u> (syntax below)	<u>BREAKPOINT BP</u>	Deletes breakpoints as described in the relevant section in Breakpoint Maintenance.
	<u>SPY</u>	Deletes breakpoints and watchpoints: see also Delete Spy in Spy Maintenance.
	<u>WATCHPOINT WP</u>	Deletes watchpoints as described in the relevant section in Watchpoint Maintenance.
	<u>ENVIRONMENT</u>	Deletes the specified debug environment: see also Delete Debug Environment in Debug Environment Maintenance.
<u>DISPLAY</u> (syntax below)	<u>BREAKPOINT BP</u>	Displays breakpoints as described in the relevant section in Breakpoint Maintenance.
	<u>SPY</u>	Displays breakpoints and watchpoints: see also Display Spy in Spy Maintenance.
	<u>WATCHPOINT WP</u>	Displays watchpoints as described in the relevant section in Watchpoint Maintenance.
	<u>CALL</u>	Displays statistics on programming objects invoked during the execution of an application: see also Display Called Objects in Call Statistics Information.
	<u>EXEC</u>	Displays statistics on executed statement lines of invoked programming objects: see also Display Executed Statement Lines in Statement Execution Statistics.
	<u>HEXADECIMAL</u>	Displays the contents of variables in hexadecimal format.
	<u>NOCALL</u>	Displays statistics on programming objects that have not been invoked during the execution of an application: see also Display Non-Called Objects in Call Statistics Information.
	<u>NOEXEC</u>	Displays statistics on non-executed statement lines of invoked programming objects: see also Display Non-Executed Statement Lines in Statement Execution Statistics.
	<u>OBJECT</u>	Displays statistics on the call frequency of objects: see also Display All Objects in Call Statistics Information.
	<u>STATEMENT</u>	Display statistics on executed and non-executed statement lines of invoked programming objects: see Display All Statement Lines in Statement Execution Statistics.
	<u>VARIABLE</u>	Displays variables for interrupted objects as described in the relevant section in Variable Maintenance.
<u>XSTATISTICS</u>	Displays a statistical summary of execution statistics: see also Display Statement Execution Statistics in Statement Execution Statistics.	
<u>ESCAPE</u>	<u>BOTTOM</u>	Stops processing a loop and escapes to the first statement after the loop: see ESCAPE BOTTOM in Execution Control Commands.
	<u>ROUTINE</u>	Stops processing an interrupted object and continues with another object, if available: see ESCAPE ROUTINE in Execution Control Commands.
<u>EXIT</u>		Leaves the current screen: see EXIT in Execution Control Commands.
<u>FLIP</u>		Switches between the display of the two PF-key lines (PF1 to PF12 and PF13 to PF24).

Command	Subcommand(s)	Explanation
<u>G</u> O		Returns control to the execution of the interrupted program: see GO in Execution Control Commands.
LAST		Displays the command entered last. The last three commands are stored and can be recalled.
<u>L</u> EFT		Shifts to the left side of a source code listing.
<u>L</u> IST		Displays the source code of a object.
(syntax below)	<u>B</u> REAK	Shows the object source with the current break. The relevant statement line is highlighted.
	<u>L</u> ASTLINE	Shows the object source with the last line executed before the current break.
<u>L</u> OAD	<u>E</u> NVIRONMENT	Loads the debug environment specified: see Load Debug Environment in Debug Environment Maintenance.
(syntax below)		
<u>M</u> ENU		Invokes the Debug Main Menu.
<u>M</u> ODIFY	<u>B</u> REAKPOINT	Modifies breakpoints as described in the relevant section in Breakpoint Maintenance.
(syntax below)	<u>B</u> P	
	<u>S</u> PY	Invokes the Modify Breakpoint or Modify Watchpoint screen: see also Modify Spy in Spy Maintenance.
	<u>W</u> ATCHPOINT	Modifies watchpoints as described in the relevant section in Watchpoint Maintenance.
	<u>W</u> P	
	<u>H</u> EXADECIMAL	Modifies the contents of variables in hexadecimal format.
	<u>V</u> ARIABLE	Invokes the Display Variable screen for modification as described in the relevant section in Variable Maintenance. See also Modify Variable.
<u>N</u> EXT		Executes the next command specified for a breakpoint or watchpoint.
<u>O</u> BJCHAIN		Displays executed objects at various program levels: see OBJCHAIN in Navigation and Information Commands.
ON/OFF		Switches test mode on/off.

Command	Subcommand(s)	Explanation
<u>P</u> RINT (syntax below)	<u>C</u> ALL	Prints statistics on programming objects invoked during the execution of an application: see also Display Called Objects in Call Statistics Information.
	<u>E</u> XC	Prints statistics on executed statement lines of invoked programming objects: see also Display Executed Statement Lines in Statement Execution Statistics.
	<u>N</u> O <u>C</u> ALL	Prints statistics on programming objects that have not been invoked during the execution of an application: see also Display Non-Called Objects in Call Statistics Information.
	<u>N</u> O <u>E</u> XEC	Prints statistics on non-executed statement lines of invoked programming objects: see also Display Non-Executed Statement Lines in Statement Execution Statistics.
	<u>O</u> BJECT	Prints statistics on the call frequency of objects: see also Display All Objects in Call Statistics Information.
	<u>S</u> TATEMENT	Prints statistics on executed and non-executed statement lines of invoked programming objects: see also Display All Statement Lines in Statement Execution Statistics.
	<u>X</u> STATISTICS	Prints statistics on executed statement lines: see also Display Statement Execution Statistics in Statement Execution Statistics.
<u>P</u> ROFILE		Displays the Edit Profile screen where you can modify the profile of the Natural Debugger. For details on the Edit Profile screen, see the relevant section in Navigation and Information Commands.
<u>R</u> ESET (syntax below)	<u>E</u> NVIRONMENT	Resets the current debug environment: see Reset Debug Environment in Debug Environment Maintenance.
<u>R</u> IGHT		Shifts to the right side of a source code listing.
<u>R</u> UN		Switches off test mode and continues program execution.
<u>S</u> AVE (syntax below)	<u>E</u> NVIRONMENT	Resets the current environment and saves the debug specifications. See also Save Debug Environment in Debug Environment Maintenance.
<u>S</u> CAN	<u>A</u> BS	Only applies to the List Object Source (see the relevant section) function. Searches for a value in the source code of an object: see SCAN in Navigation and Information commands and Syntax Diagrams below.
<u>S</u> CREEN		When entered upon interruption of an object, the current screen output of the interrupted object is displayed. ENTER takes you back to debug mode.

Command	Subcommand(s)	Explanation
<u>SET</u> (syntax below)	<u>BREAKPOINT</u> <u>BP</u>	Invokes the Set Breakpoint menu.
	CALL ON/OFF	Activates/deactivates the Call Statistics Maintenance as described in the relevant section.
	<u>OBJECT</u>	Changes the default object defined for the Natural debugging utility. See also SET OBJECT in Navigation and Information.
	<u>WATCHPOINT</u> <u>WP</u>	Invokes the Set Watchpoint menu.
	<u>XSTATISTICS</u> ON/OFF/COUNT	Activates the statistics function about executed statements of programming objects. See also Set Statement Execution Statistics in Statement Execution Statistics.
SM		Invokes the Spy Maintenance menu as described in the relevant section.
<u>STACK</u>		Displays the contents of the entry at the top of the Natural stack: see STACK in Navigation and Information Commands.
<u>STEP</u>	[n]	Continues an interrupted object for <i>n</i> statement lines. The default value for <i>n</i> is 1.
	<u>SKIPSUBLEVEL</u>	Continues step-mode processing of interrupted objects without entering programs at sub-levels: see also SKIPSUBLEVEL in Execution Control Commands.
<u>STOP</u>		Terminates both the Natural Debugger and any interrupted Natural object; the NEXT line is displayed.
<u>SYSVARS</u>		Displays the current values of system variables (except the database-related system variables). See also Display Variables in Variable Maintenance.
<u>TEST ON/OFF</u>		Switches test mode on/off. Can also be entered at NEXT command level.
WM		Invokes the Watchpoint Maintenance menu as described in the relevant section.

Syntax Diagrams

The syntax diagrams listed below refer to more complex command sequences.

For a detailed explanation of the symbols used within the syntax descriptions, see the section System Command Syntax in the Natural Command Reference documentation.

For better readability, synonymous keywords are omitted from the syntax diagrams below. An underlined portion of a keyword represents an acceptable abbreviation.

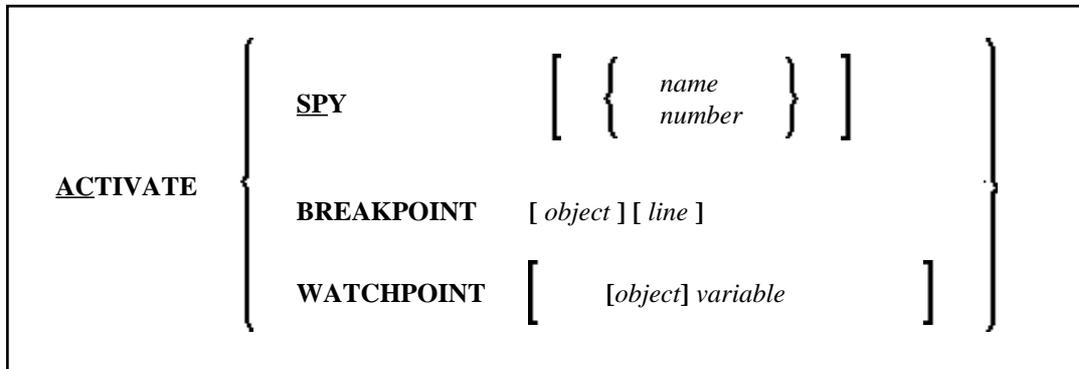
Valid synonyms are:

Keyword	Synonym
BREAKPOINT	BP
DEACTIVATE	DA
WATCHPOINT	WP

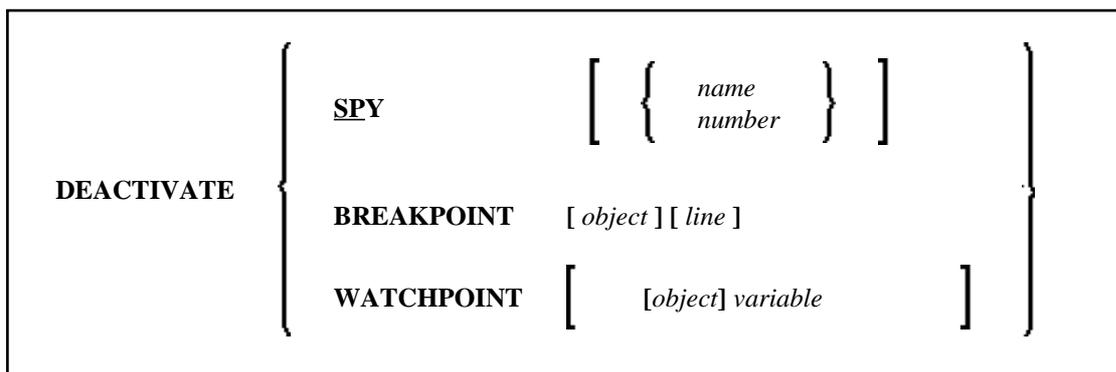
Below is information on:

- ACTIVATE
- DEACTIVATE
- DELETE
- DISPLAY
- LIST
- LOAD
- MODIFY
- PRINT
- RESET
- SAVE
- SET

ACTIVATE



DEACTIVATE



DELETE

<u>DELETE</u>	<u>SPY</u> [{ <i>name</i> <i>number</i> }]	
	BREAKPOINT	[<i>object</i>] [<i>line</i>]
	WATCHPOINT	[[<i>object</i>] <i>variable</i>]
	<u>XSTATISTICS</u>	[[<i>library</i>] <i>object</i>]
	<u>ENVIRONMENT</u>	[<i>name</i>]

DISPLAY

<u>DISPLAY</u>	<u>SPY</u> [{ <i>name</i> <i>number</i> }]	
	BREAKPOINT	[<i>object</i>] [<i>line</i>]
	WATCHPOINT	[[<i>object</i>] <i>variable</i>]
	CALL	
	<u>OBJECT</u>	
	<u>NOCALL</u>	
	<u>XSTATISTICS</u>	<i>library</i> [<i>object</i>]
	STATEMENT	
	<u>EXEC</u>	
	<u>NOEXEC</u>	
	<u>VARIABLE</u>	[<i>variable</i> ,...]
	<u>HEXADECIMAL</u>	

LIST

<u>LIST</u>	{	<u>LASTLINE</u> <u>BREAK</u> <i>object [line]</i>	}
-------------	---	---	---

LOAD

<u>LOAD ENVIRONMENT</u> [<i>name</i>]

MODIFY

<u>MODIFY</u>	{	<u>SPY</u> [{ <i>name</i> <i>number</i> }]	}
		<u>BREAKPOINT</u> [<i>object</i>] [<i>line</i>]	
		<u>WATCHPOINT</u> [[<i>object</i>] <i>variable</i>]	
		<u>VARIABLE</u> [<i>variable</i> [= <i>new value</i>]]	
		<u>HEXADECIMAL</u>	

PRINT

<u>PRINT</u>	{	<u>CALL</u> <u>OBJECT</u> <u>NOCALL</u> <u>XSTATISTICS</u> <u>STATEMENT</u> <u>EXEC</u> <u>NOEXEC</u>	}	<i>library</i> [<i>object</i>]
--------------	---	---	---	----------------------------------

RESET

RESET ENVIRONMENT [*name*]

SAVE

SAVE ENVIRONMENT [*name*]

SET

	OBJECT	<i>object</i>			
SET	BREAKPOINT	[<i>object</i>]		[{ <i>line label</i> }]	
	WATCHPOINT	[[<i>object</i>]		<i>variable</i>]	
	CALL	{ OFF			
		{ ON			
		}			
	XSTATISTICS	{ OFF			
		{ ON		[<i>library</i> [<i>object</i>]	}
		{ COUNT			}
		}			}

DBLOG Utility - Logging Database Calls

The DBLOG utility is used to log Adabas commands, DL/I (and SYNC/ROLB) calls, or SQL statements. Logging is useful for tuning an application (controlling the flow of commands accessing the database) and for analysing error codes that may be returned from the database.

Related Documentation:

Adabas, DL/I and DB2.

This section covers the following topics:

- Executing DBLOG
- DBLOG Menu
- DBLOG Trace Screen
- DBLOG Snapshot Function
- DBLOG Direct Commands

Executing DBLOG

The DBLOG utility logs each Adabas command, DL/I (and SYNC/ROLB) calls or SQL statement after it has been processed by the database system. Log recording starts when you activate DBLOG and execute or run a Natural program. Logging remains active until you enter an appropriate command or until you terminate your Natural session.

Below is information on:

- Data Processing and Storage
 - Activating and Deactivating DBLOG
 - Using Selective DBLOG
-

Data Processing and Storage

The data logged by the DLBOG utility are written into the Natural debug buffer. The size of the buffer is determined by the profile parameter DSIZE as described in the Natural Parameter Reference documentation. If there is not enough space available, only the most recent log data will be held in the Natural debug buffer.

DBLOG can be used online or in batch mode. DL/I and SYNC/ROLB calls can be logged under CICS, under IMS/TM or in batch mode.

For further details on batch-mode processing, refer to Natural in Batch Mode as described in the Natural Operations for Mainframes documentation.

The logs recorded are displayed on the DBLOG Trace screen.

The DBLOG utility provides default settings for data recording. With the DBLOG Menu, you can specify selection criteria for the commands, calls or statements to be logged and the information displayed. The DBLOG Menu also provides functions for activating or deactivating logging. To control DBLOG execution, you can also use direct commands.

The fields of the DBLOG Trace screen, the DBLOG Menu and direct command execution are explained in the relevant sections of the DBLOG documentation.

Activating and Deactivating DBLOG

Below are the commands that apply when invoking DBLOG using the default settings for logging records. See also DBLOG Direct Commands for additional information.

To activate or deactivate DBLOG

- Adabas:
In the command line, enter the toggle command TEST DBLOG.
Or, in the command line, enter TEST DBLOG ON or TEST DBLOG OFF.
Or, on the DBLOG Menu, enter Function Code **B** or Function Code **E**.
- DL/I:
In the command line, enter the toggle command TEST DBLOG D.
Or, in the command line, enter TEST DBLOG D ON or TEST DBLOG D OFF.
Or, on the DBLOG Menu, enter Function Code **B** or Function Code **E**.
- SQL:
In the command line, enter the toggle command TEST DBLOG Q.
Or, in the command line, enter TEST DBLOG Q ON or TEST DBLOG Q OFF.

Or, on the DBLOG Menu, enter Function Code **B** or Function Code **E**.

Using Selective DBLOG

Below is an example of logging Adabas commands, DL/I calls or SQL statements with selection criteria specified with the DBLOG Menu.

To perform DBLOG with selection criteria

1. Invoke the DBLOG Menu:
 - Adabas:
In the command line, enter TEST DBLOG MENU.
 - DL/I:
In the command line, enter TEST DBLOG D MENU.
 - SQL:
In the command line, enter TEST DBLOG Q MENU.
2. On the DBLOG Menu, specify logging restrictions and activate logging:
Complete the input fields and enter Function Code **B**.
The message "DBLOG started now" is displayed.
3. Execute a Natural program which contains Adabas commands, DL/I calls or SQL statements.
4. Invoke the DBLOG Trace screen and deactivate logging:
 - Adabas:
In the command line, enter TEST DBLOG.
 - DL/I:
In the command line, enter TEST DBLOG D.
 - SQL:
In the command line, enter TEST DBLOG Q.

The DBLOG Trace screen appears.
5. Clear the Natural debug buffer and deactivate logging:
 - Adabas:
In the command line, enter TEST DBLOG OFF.
 - DL/I:
In the command line, enter TEST DBLOG D OFF.
 - SQL:
In the command line, enter TEST DBLOG Q OFF.

DBLOG terminates and the NEXT line appears.

See also DBLOG Direct Commands for additional information.

DBLOG Menu

On the DBLOG Menu, you can activate or deactivate logging and specify which Adabas commands, DL/I calls or SQL statements are to be logged.

To invoke the DBLOG Menu

- Adabas:
In the command line, enter TEST DBLOG MENU.
- DL/I:
In the command line, enter TEST DBLOG D MENU.
- SQL:
In the command line, enter TEST DBLOG Q MENU.

Below is information on:

- DBLOG Menu - Functions
 - DBLOG Menu - Specifying Restrictions
-

DBLOG Menu - Functions

To execute the functions provided on the DBLOG Menu, enter a code in the relevant input field or choose a PF key:

Code or PF Key	Function	Explanation
B PF4/Begin	Begin Logging of ...	Activates the DBLOG logging of the Adabas commands, DL/I calls or SQL statements that match the selection criteria. See also alternative commands in DBLOG Direct Commands.
	Optional Buffers for Code B	Only applicable to Adabas commands. Selects additional Adabas buffers to be logged: see Specifying Buffers below.
E PF5/End	End and Display Log Records	Deactivates logging and displays the DBLOG Trace screen of the current log record if data exists in the Natural debug buffer. Current log data are kept in the Natural debug buffer. See also alternative commands in DBLOG Direct Commands.
S PF6/Snap	Snapshot of Specific...	Adabas: Interrupts a program at a specified Adabas command and displays detailed information on this command only: see Snapshot Function for Adabas commands. DL/I and SQL: Collects detailed information on a specified DL/I call or SQL statement: see Snapshot Function for DL/I calls and SQL statements.
PF3/Exit		Leaves the DBLOG Menu and returns to the NEXT line. The current log records are kept in the Natural debug buffer.
PF12/Canc		Clears the Natural debug buffer, leaves the DBLOG Menu and returns to the NEXT line.

DBLOG Menu - Specifying Restrictions

Below are the input fields the DBLOG Menu provides for specifying selection criteria to restrict logging:

Field	Explanation
Skip	Only applicable with Function Code S. Number of commands, calls or statements to be skipped before logging is to start.
Program	Restricts logging to commands, calls or statements issued by the program specified.
DBID	Only applicable to Adabas commands. Restricts logging to commands issued for the database ID specified.
FNR	Only applicable to Adabas commands. Restricts logging to commands issued for the file number specified.
Line from Line to	Restricts logging to commands, calls or statements within the range of the source line numbers specified.
Low Resp High Resp	Only applicable to Adabas commands. Restricts logging to commands which result in a response code within the range specified.
Low Stat High Stat	Only applicable to DL/I calls. Restricts logging to calls which result in a status code within the range specified.
Low SQLC High SQLC	Only applicable to SQL statements. Restricts logging to statements which result in an SQL return code within the range specified.

Specifying Buffers

Only applicable to Adabas commands.

In addition to the Adabas control block (CB) which is logged by default, you can select for logging one or more of the following Adabas buffers by entering any character in the input field next to the buffer(s) desired:

FB	Format Buffer
RB	Record Buffer
SB	Search Buffer
VB	Value Buffer
IB	ISN Buffer

The logs of the buffers (maximum is 80 bytes per buffer) can be displayed on the DBLOG Trace screen as described in the relevant section under Displaying Buffers.

The snapshot function (see the relevant section) logs all Adabas buffers by default. Therefore, you need not mark any of the optional buffers before you execute this function.

DBLOG Trace Screen

The DBLOG Trace screen displays recorded log data on Adabas commands, DL/I calls or SQL statements which are kept in the Natural debug buffer.

Below is information on the DBLOG Trace screen for

- Adabas Commands
- DL/I Calls
- SQL Statements

DBLOG Trace Screen - Adabas Commands

- Invoking DBLOG Trace - Adabas Commands
- DBLOG Trace Adabas Commands - Fields and Functions
- Displaying Buffers on the DBLOG Trace Screen

Invoking DBLOG Trace - Adabas Commands

Below is an example of invoking the DBLOG Trace screen for Adabas commands.

1. Write the following Natural program in reporting mode:

```
/* PROG1
READ (3) EMPLOYEES BY NAME
DISPLAY NAME
END
```

2. In the command line, enter TEST DBLOG.
The message "DBLOG started now" is displayed.
3. In the command line, enter RUN.
4. In the command line, enter TEST DBLOG again.

Logging is deactivated and the DBLOG Trace screen appears as shown in the example below:

16:22:50		***** NATURAL TEST UTILITIES *****							2002-03-11			
User SAG		- DBLOG Trace -							Library SAG			
M	No	Cmd	DB	FNR	Rsp	ISN	ISQ	CID	CID(Hex)	OP	Pgm	Line
—	1	L3	10	316		1300		???	00200101	V		0020
—	2	L3	10	316		1530		???	00200101	V		0020
—	3	L3	10	316		478		???	00200101	V		0020
—	4	RC	10	316				???	00200101	SI		0020
—	5	RC	10						00000000	F		0040
—												
—												
Command ==>												
Enter-PF1---PF2---PF3---PF4---PF5---PF6---PF7---PF8---PF9---PF10---PF11---PF12---												
Help Print Exit Top Posi Bot - + Canc												

DBLOG Trace Adabas Commands - Fields and Functions

The fields and functions provided on the DBLOG Trace screen for Adabas commands are shown below:

Column	PF Key	Explanation
M		Input option for line commands that invoke extra windows with detailed information on Adabas buffers: see Displaying Buffers below.
No		Sequence number. The commands are displayed in the sequence in which they were executed.
Cmd		Adabas command.
DB		Database ID.
FNR		File number.
Rsp		Adabas response code.
ISN		Internal sequence number of record.
ISQ		ISN quantity.
CID		Command ID.
CID (Hex)		Command ID in hexadecimal format.
OP		Adabas Command Options 1 and 2.
Pgm		Program name.
Line		Source code line number.
	PF2 Print	Prints a hardcopy of a screenshot.
	PF3 Exit	Leaves the DBLOG Trace screen and returns to the NEXT line. The current log records are kept in the Natural debug buffer.
	PF4 Top	Goes to the top of the list.
	PF5 Posi	Moves log entries to the top of the screen: In Column M , position the cursor next to the desired command and sequence number listed in column No and choose PF5. The logs are repositioned starting with the sequence number selected.
	PF6 Bot	Goes to the bottom of the list.
	PF7 -	Scrolls up one page in the list.
	PF8 +	Scrolls down one page in a list.
	PF12 Canc	Clears the Natural debug buffer and deactivates logging.

Displaying Buffers on the DBLOG Trace Screen

To display detailed buffer information, except for the Adabas control block which is recorded by default, you need to specify the buffer(s) desired in the DBLOG Menu screen (see the relevant sections) when logging is started. Otherwise, no logs will be recorded for the corresponding buffer(s). If, for example, only logging of the format buffer has been specified in the DBLOG Menu, you can only display the Format Buffer window but not the Record Buffer window. For each buffer specified, only the first 80 bytes are logged.

To display detailed buffer information

- On the DBLOG Trace screen, in the input field, next to the log(s) desired, enter any of the following line commands that corresponds to the buffer desired:

C	Control Block
F	Format Buffer
R	Record Buffer
S	Search Buffer
V	Value Buffer
I	ISN Buffer

An extra window displays the buffer specified. Below are example screens of the control block and the format buffer.

Example Screen of Control Block:

```

16:31:20          ***** NATURAL TEST UTILITIES *****          2002-03-11
User SAG          - DBLOG Trace -          Library SAG
M  No Cmd  DB   FNR  Rsp      ISN      ISQ  CID  CID(Hex)  OP  Pgm      Line
C   1  L3   10   316      1300      ??? 00200101  V  SAGTEST  0020
_   2  L3   10   316      1530      ??? 00200101  V  SAGTEST  0020
- +-----+-----+-----+-----+-----+-----+-----+-----+
- !  _  SEQ NO ..      1  PROGRAM .. SAGTEST  LINE .. 0020          !
- ! Command Code .. L3      Command ID ... ??? 00200101  CB Start ... 30D5 !
- ! Response Code . 0000      ISN .....      1300      FNR first .. 01  !
- ! ISN Low Limit .          ISN Quantity .          FNR last ... 3C  !
- ! FB Length ..... 0009      RB Length .... 0014          SB Length .. 0008 !
- ! VB Length ..... 0014      IB Length .... 0000          !
- ! Command Opts ..  V          !
- ! Additions 1 ... AE]?      C1C5BBCA40404040      Command Time .. 00000009 !
- ! Additions 2 ...  ? ?      00120014          User Area ..... 00000000 !
- !          !
- ! Additions 3 ...          0000000000000000          !
- ! Additions 4 ...          0000000000000000          !
- ! Global FID .... 0000000000000000          !
- +-----+-----+-----+-----+-----+-----+
Command ==>
Enter-PF1---PF2---PF3---PF4---PF5---PF6---PF7---PF8---PF9---PF10--PF11--PF12---
      Help Print Exit Top  Posi Bot  -  +          Canc
    
```

Example Screen of Format Buffer:

M	No	Cmd	DB	FNR	Rsp	ISN	ISQ	CID	CID(Hex)	OP	Pgm	Line
f	1	L3	10	316		1300		???	00200101	V	SAGTEST	0020
—	2	L3	10	316		1530		???	00200101	V	SAGTEST	0020
+-----+ 0												
—	!	—	Seq No	..	1	Format Buffer						! 0
—	!	0000	*	C1C56BF0	F2F06BC1	4B000000	00000000	*	AE,020,A.			* 0000 ! 0
—	!	0010	*	00000000	00000000	00000000	00000000	*				* 0010 !
—	!											!
—	!	0020	*	00000000	00000000	00000000	00000000	*				* 0020 !
—	!	0030	*	00000000	00000000	00000000	00000000	*				* 0030 !
—	!	0040	*	00000000	00000000	00000000	00000000	*				* 0040 !
+-----+ 0												

DBLOG Trace Screen - DL/I Calls

- Invoking DBLOG Trace - DL/L Calls
- DBLOG Trace DL/L Calls - Fields and Functions

Invoking DBLOG Trace - DL/L Calls

Below is an example of invoking the DBLOG Trace screen for DL/I calls.

1. Write the following Natural program:

```

DEFINE DATA LOCAL
01 COURSE VIEW OF DNDL01-COURSE
  02 COURSEN (A3)
  02 TITLE (A33)
01 OFFERING VIEW OF DNDL01-OFFERING
  02 COURSEN-COURSE (A3)
  02 LOCATION (A31)
END-DEFINE
READ (5) COURSE BY COURSEN
  IF TITLE = 'NATURAL'
    FIND (1) OFFERING WITH COURSEN-COURSE = COURSEN
      MOVE 'DARMSTADT' TO LOCATION
      UPDATE
    END OF TRANSACTION
  END-FIND
END-IF
END-READ
END

```

2. In the command line, enter TEST DBLOG D.
The message "DBLOG started now" is displayed.
3. In the command line, enter RUN.
4. In the command line, enter TEST DBLOG D again.
Logging is deactivated and the DBLOG Trace screen for DL/I screen is displayed:

```

08:13:15          ***** NATURAL DBA Utility *****          2002-03-12
User SAG          - DBLOG Trace -          Library SAG
No  Func PCB NS SC DBD/PSB      First SSA (truncated)      IOA (trunc) Program  Line
-----
 1 PCB          PANII41          SAGTEST 0090
 2 GU   1   1   DNDL01  COURSE *--(COURSESEN => .          SAGTEST 0090
 3 GN   1   1   DNDL01  COURSE *--(COURSESEN => .Z01        SAGTEST 0090
 4 GN   1   1   DNDL01  COURSE *--(COURSESEN => .001        SAGTEST 0090
 5 GN   1   1   DNDL01  COURSE *--(COURSESEN => .004NATURA SAGTEST 0090
 6 GHNP 1   2   DNDL01  COURSE *- (COURSESEN =004 ?010791DAR SAGTEST 0110
 7 REPL 1          DNDL01          ?010791DAR SAGTEST 0130
 8 SYNC
 9 PCB          PANII41          SAGTEST 0110
10 GU   1   1   DNDL01  COURSE *--(COURSESEN = 004 .004NATURA SAGTEST 0110
11 GHNP 1   2   DNDL01  COURSE *--(COURSESEN = 004 ?010791DAR SAGTEST 0110
12 GN   1   1   DNDL01  COURSE *--(COURSESEN => +110        SAGTEST 0090
***** End of Log *****
    
```

DBLOG Trace DL/I Calls - Fields and Functions

Following are the fields provided on the DBLOG Trace screen for DL/I calls. For an explanation of the PF keys provided, refer to the description of the DBLOG Trace menu for Adabas above.

Column	Explanation
No	Sequence number. The calls are displayed in the sequence in which they were executed.
Func	DL/I function.
PCB	PCB number.
NS	Number of SSAs.
SC	DL/I status code.
DBD/PSB	DBD name for DB calls. PSB name for scheduling calls.
First SSA	First 25 bytes of the first SSA.
IOA	First 13 bytes of the I/O area.
Program	Natural program name.
Line	Source-code line number.

DBLOG Trace Screen - SQL Statements

- Invoking DBLOG Trace - SQL Statements
- DBLOG Trace SQL Statements - Fields and Functions

Invoking DBLOG Trace - SQL Statements

Below is an example of the Trace Function for SQL statements.

1. Write the following Natural program:

```

DEFINE DATA LOCAL
01 EMP VIEW OF DSN8510-EMP
02 EMPNO
02 FIRSTNME
    
```

```

02 MIDINIT
02 LASTNAME
02 EDLEVEL
02 SALARY
01 EMPPROJACT VIEW OF DSN8510-EMPPROJACT
02 EMPNO
02 PROJNO
02 ACTNO
02 EMPTIME
END-DEFINE
FIND (1) EMP WITH EMPNO> '000300'
  FIND (1) EMPPROJACT WITH EMPNO = EMPNO(0150)
  MOVE 0.75 TO EMPTIME
  UPDATE
END-FIND
ADD 1 TO EDLEVEL
UPDATE
END-FIND
*
FIND (1) EMP WITH EMPNO> '000300'
  FIND (1) EMPPROJACT WITH EMPNO = EMPNO(0240)
  DISPLAY EMPPROJACT EMP.EDLEVEL
END-FIND
END-FIND
ROLLBACK
END
    
```

2. In the command line, enter TEST DBLOG Q.
The message "DBLOG started now" is displayed.
3. In the command line, enter RUN.
4. In the command line, enter TEST DBLOG Q again.
Logging is deactivated and the DBLOG Trace screen for SQL statements is displayed:

16:24:27		***** NATURAL Test Utilities *****							2002-04-03			
User SAG		- DBLOG Trace -							Library SAG			
M No	R	SQL Statement (truncated)	CU	SN	SREF	M	Typ	SQLC/W	Program	Line	LV	
--	1	SELECT EMPNO,FIRSTNME,MIDINIT	01	01	0150	D	DB2		SAGTEST	0150	01	
--	2	FETCH CURSOR	01	01	0150	D	DB2		SAGTEST	0150	01	
--	3	SELECT EMPNO,PROJNO,ACTNO,EMP	02	02	0160	D	DB2		SAGTEST	0160	01	
--	4	FETCH CURSOR	02	02	0160	D	DB2		SAGTEST	0160	01	
--	5	UPDATE DSN8510.EMPPROJACT SET	02	03	0160	D	DB2		SAGTEST	0180	01	
--	6	CLOSE CURSOR	02	02	0160	D	DB2		SAGTEST	0160	01	
--	7	UPDATE DSN8510.EMP SET EDLEVE	01	04	0150	D	DB2		SAGTEST	0210	01	
--	8	CLOSE CURSOR	01	01	0150	D	DB2		SAGTEST	0150	01	
--	9	SELECT EMPNO,FIRSTNME,MIDINIT	05	05	0240	D	DB2		SAGTEST	0240	01	
--	10	FETCH CURSOR	05	05	0240	D	DB2		SAGTEST	0240	01	
--	11	SELECT EMPNO,PROJNO,ACTNO,EMP	06	06	0250	D	DB2		SAGTEST	0250	01	
--	12	FETCH CURSOR	06	06	0250	D	DB2		SAGTEST	0250	01	
--	13	CLOSE CURSOR	06	06	0250	D	DB2		SAGTEST	0250	01	
--	14	CLOSE CURSOR	05	05	0240	D	DB2		SAGTEST	0240	01	
--	15	ROLLBACK	00	00	0000	D	DB2		SAGTEST	0290	01	
--												
--												
Command ==>												
Enter-PF1---		PF2---	PF3---	PF4---	PF5---	PF6---	PF7---	PF8---	PF9---	PF10--	PF11--	PF12---
Help		Print	Exit	Top	Posi	Bot	-	+				Canc

DBLOG Trace SQL Statements - Fields and Functions

The fields provided on the DBLOG Trace screen for SQL statements are shown below. For an explanation of the PF keys provided, refer to the description of the DBLOG Trace menu for Adabas above.

Column	Explanation
M	<p>Input option for line commands:</p> <p>E Executes the command EXPLAIN which provides information on the DB2 or SQL/DS optimizer's choice of strategy for executing SQL statements.</p> <p>See also EXPLAIN Command in: Natural System Commands for DB2, Natural for DB2 documentation. Database Management, Natural for SQL/DS documentation.</p> <p>L Executes the command LISTSQL which lists the Natural statements in the source code of an object and the corresponding SQL statements into which they have been translated. An SQL statement is identified by the library name, program name, and line number taken from the Natural debug buffer.</p> <p>See also LISTSQL Command in: Natural System Commands for DB2, Natural for DB2 documentation. Database Management, Natural for SQL/DS documentation.</p> <p>Important: Since both commands obtain their information from the Natural system file, unwanted results may occur if the corresponding Natural program has been recataloged after the logging function was executed with the TEST DBLOG Q command. These unwanted results may be caused by statements modified after the logging.</p>
No	Sequence number; the statements are displayed in the sequence in which they were executed.
R	<p>Only applicable if the Natural File Server for DB2 is in use.</p> <p>Indicates by an asterisk in front of the corresponding statement that a reselection has been performed; if not, the column is left blank.</p> <p>See also Concept of the File Server in Natural File Server for DB2 (Natural for DB2 documentation).</p>
SQL Statement	The first 29 characters of the logged SQL statement.
CU	Cursor number.
SN	Internal statement number.
SREF	Statement reference number.
M	Mode: D for dynamic or S for static.
Typ	Database type: DB2 or /DS.
SQLC/W	Either the SQL return code in the SQLCODE field of the SQLCA, or the warning in the SQLWARN0 field of the SQLCA if SQLCODE is 0.

Column	Explanation
Pgm	Natural program name.
Line	Source code line number.
LV	Program level.

DBLOG - Snapshot Function

The snapshot function provides detailed information on one particular Adabas command, DL/I call or SQL statement.

Below is information on the snapshot function for

- Adabas Commands
 - DL/I Calls
 - SQL Statements
-

Snapshot Function - Adabas Commands

This snapshot function interrupts program execution after executing the first Adabas command that matches the selection criteria specified on the DBLOG Menu. The Snapshot Report (example below) generated for the specified Adabas command is displayed immediately after program interruption.

The snapshot function automatically logs **all** Adabas buffers. Therefore, you do not have to mark any of the optional buffers on the DBLOG Menu before you start the snapshot function. The default Snapshot Report displays the Adabas control block (CB).

Below is information on:

- Invoking Snapshot - Adabas Commands
- Displaying Buffers on the Snapshot Report

Invoking Snapshot - Adabas Commands

To invoke the Snapshot Report screen for Adabas commands

- On the DBLOG Menu, specify an Adabas command and additional criteria, if desired, and enter Function Code **S**.
The message "DBLOG snapshot facility started now" is displayed.
- Execute a Natural program which contains the Adabas command specified on the DBLOG Menu.
The program stops executing and the Snapshot Report for Adabas commands is displayed as shown in the example below:

```

16:36:39          ***** NATURAL TEST UTILITIES *****          2002-03-11
                    - Snapshot Report -

Command Code : L3          Command ID   : ??? 00200101 File Number : 013C
Response Code:      0      ISN          :          1300
ISN Low Limit: 00000000   ISN Quantity:          0
FB Length   : 0009       RB Length    : 0014          SB Length   : 0008
VB Length   : 0014       IB Length    : 0000          Com. Option 1:
Com. Option 2: V         Additions 1  : AE]?          Additions 2  : ? ?
Additions 3  :           Additions 4  :
Global FID   : 0000000000000000 Command Time : 00000019 Pgm: SAGTEST Lin: 0020
Control Block
0000 * 30D5D3F3 00200101 013C0000 00000514 * ?NL3 ?????   ?? * 0000
0010 * 00000000 00000000 00090014 00080014 *           ? ? ? ? * 0010
0020 * 000000E5 C1C5BBCA 40404040 00120014 *   VAE]?       ? ? * 0020
0030 * 00000000 00000000 00000000 00000000 *                   * 0030
0040 * 00000000 00000000 00000019 00000000 *                   ?   * 0040
0050 * 00000000 00000000 00000000 00000000 *                   * 0050
0060 * 00000000 00000000 00000000 00000000 *                   * 0060
0070 * 00000000 00000000 00000000 00000000 *                   * 0070

Command ==> CB
Enter-PF1---PF2---PF3---PF4---PF5---PF6---PF7---PF8---PF9---PF10---PF11---PF12---
      Help      Exit  CB   FB   RB   -   +   SB   VB   IB   Canc
    
```

Displaying Buffers on the Snapshot Report

The Snapshot Report shows the Adabas control block (CB) by default. To display different Adabas buffers and page up or down the screen, choose any of the direct commands or PF keys listed below:

PF Key	Direct Command	Buffer
PF4	CB	Displays the Control Block. This is the default.
PF5	FB	Displays the Format Buffer.
PF6	RB	Displays the Record Buffer.
PF7	-	Scrolls up the screen to display long buffers logs that extend beyond the terminal screen.
PF8	+	Scrolls down the screen to display long buffers logs that extend beyond the terminal screen.
PF9	SB	Displays the Search Buffer.
PF10	VB	Displays the Value Buffer.
PF11	IB	Displays the ISN Buffer.

Snapshot Function - DL/I Calls

This snapshot function generates the Snapshot Report (example below) of the first DL/I call that matches the selection criteria specified on the DBLOG Menu. A snapshot does not interrupt the program flow. The snapshot data are kept in the Natural debug buffer to be displayed only if the user enters the appropriate DBLOG command as described below.

Below is information on:

- Status Code
- Processing Options
- Segment Name
- Length of the Key Feedback Area
- Number of Sensitive Segments
- Key Feedback Area
- Number of SSAs (Segment Search Argument)
- all SSAs
- the I/O Area

The first 120 bytes of the Key Feedback Area, of all SSAs (up to 15 SSAs are possible) and of the I/O area are displayed, both in decimal and hexadecimal format.

The DBD Name in the PCB is used to read the corresponding NDB (Natural equivalent of DBD) from the Natural FDIC system file. In this NDB, the segment whose name is given in the PCB is located and its minimum/maximum length and segment level number are displayed. The segment level number should match the number in the PCB. In this way, it is possible to detect inconsistencies between Natural NDBs and DL/I DBDs.

The PSB name is used to read the corresponding NSB (Natural equivalent of PSB) from the Natural FDIC system file. From this NSB, the number of sensitive segments is displayed. This number should match the number in the PCB. In this way, it is possible to detect inconsistencies between Natural NSBs and DL/I PSBs.

The snapshot function checks whether the DL/I DBD/PSB and the Natural NDB/NSB contain the same values in the fields "Level Number" and "Number of SENSEGs". The same values, however, do not necessarily ensure that the DL/I DBD/PSB and the Natural NDB/NSB are fully consistent.

In the example above, the values in the "Number of SENSEGs" (Sensitive Segment Type) fields are different, because the Natural NATPSB procedure was not executed after the PSB had been changed by the DL/I PSBGEN procedure.

Snapshot Function - SQL Statements

The snapshot function generates the Snapshot Report (example below) of the first SQL statement that matches the selection criteria specified on the DBLOG Menu. A snapshot does not interrupt the program flow.

Unlike the statements displayed with the DBLOG Trace function, the snapshot shows the statement in its entirety (limited to 13 lines).

The snapshot data are kept in the Natural debug buffer to be displayed only if the user enters the appropriate DBLOG command as described below.

Below is information on:

- Invoking Snapshot - SQL Statements
- Snapshot Report Information - SQL Statements

Invoking Snapshot - SQL Statements

To invoke the Snapshot Report screen for SQL statements

- On the DBLOG Menu, specify an SQL statement and additional criteria, if desired, and enter Function Code S.
The message "DBLOG snapshot facility started now" is displayed.
- Execute a Natural program which contains the SQL statement specified on the DBLOG Menu.
(Log data are written to the Natural debug buffer.)

- Display the snapshot data:
 In the command line, enter TEST DBLOG Q.
 Or, on the DBLOG Menu, enter Function Code E.
 The Snapshot Report for SQL statements is displayed as shown in the example below:

```

10:59:28          ***** NATURAL Test Utilities *****          2002-04-08
User SAG              - Snapshot Report -              Library SAG

CU SN M Typ R SQLC/W      Library  Program  Store Clock Value   Line LV CID(Hex)
01 01 D DB2              SAG      SAGTEST  2002/04/03 14:23:06 0150 01 01500101

SQL Statement
SELECT EMPNO, FIRSTNME, MIDINIT, LASTNAME, EDLEVEL, SALARY FROM DSN8510.EMP WHERE EM
PNO> '000300' FOR UPDATE OF EDLEVEL

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

Snapshot Report Information - SQL Statements

The following information is provided on the Snapshot Report screen for SQL statements:

Item	Explanation
CU	Cursor number.
SN	Internal statement number.
M	Mode: D for dynamic or S for static.
Typ	Database type: DB2 or SQL/DS.
R	Only applicable if the Natural File Server for DB2 is in use. Indicates by an asterisk in front of the corresponding statement that a reselection has been performed; if not, the column is left blank. See also Concept of the File Server in Natural File Server for DB2 (documentation Natural for DB2).
SQLC/W	Either the SQL return code in the SQLCODE field of the SQLCA, or the warning in the SQLWARN0 field of the SQLCA if SQLCODE is 0.
Library	The library where the Natural program with the logged statement was cataloged.
Program	The name of the Natural program which contains the logged statement.
Store Clock Value	The time stamp of the Natural program which contains the logged statement.
Line	The source code line number of the logged statement.
LV	The call level of the Natural program which contains the logged statement.
CID (hex)	The command ID of the logged statement in hexadecimal format.

DBLOG Direct Commands

The direct command TEST DBLOG can be used to execute DBLOG and display or delete the log records currently stored in the Natural debug buffer. Note that TEST DBLOG does not provide any parameters to specify selection criteria. Selection criteria can only be specified with the DBLOG Menu.

The parameters that apply to the command TEST DBLOG are explained in the syntax diagrams and tables below. There are parameters that can be used to do both, activate and deactivate DBLOG (toggle effect). Activating and deactivating depends on whether or not there is data stored in the Natural debug buffer as described in Parameters below.

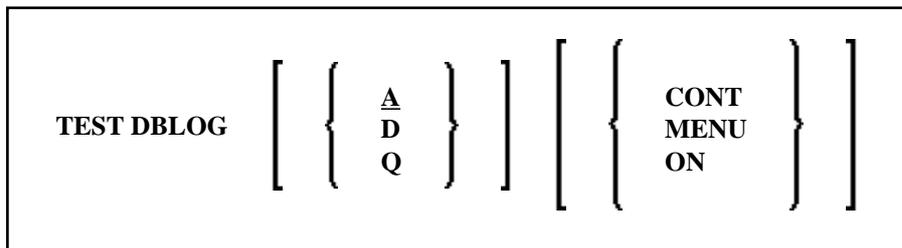
For an explanation of the symbols used in the syntax diagrams, refer to System Command Syntax in the Natural Command Reference documentation.

Below is information on:

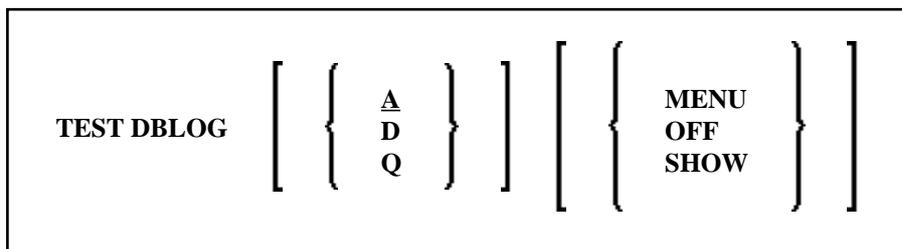
- Syntax Diagrams
- Parameters

Syntax Diagrams

Activating DBLOG



Deactivating DBLOG



Parameters

Parameter	Function
A	<p>Default value.</p> <p>Toggle function:</p> <p>Activates logging of Adabas commands if no data exist in the Natural debug buffer.</p> <p>Deactivates logging of Adabas commands and displays the DBLOG Trace screen of the current log record if data exist in the Natural debug buffer.</p>
D	<p>Toggle function:</p> <p>Activates logging of DL/I calls if no data exist in the Natural debug buffer.</p> <p>Deactivates logging of DL/I calls and displays the DBLOG Trace screen of the current log record if data exist in the Natural debug buffer.</p>
Q	<p>Toggle function:</p> <p>Activates logging of SQL statements if no data exist in the Natural debug buffer.</p> <p>Deactivates logging of SQL statements and displays the DBLOG Trace screen of the current log record if data exist in the Natural debug buffer.</p>
CONT	Activates or reactivates (restarts) logging. A restart causes DBLOG to continue logging with the next program executed or run after DBLOG execution was stopped, and to add the new logs to the data that exist from previous recordings.
MENU	Invokes the DBLOG Menu which provides the options to activate or deactivate logging and to specify the commands, calls or statements to be logged; see the relevant section.
?	
*	
SHOW	Deactivates logging and displays the DBLOG Trace screen of the current log record if data exists in the Natural debug buffer. Log record data are not deleted but kept in the Natural debug buffer.
ON	Clears the Natural debug buffer and activates logging.
OFF	Clears the Natural debug buffer and deactivates logging.

ADACALL - Issuing Adabas Direct Calls

The utility ADACALL can be used to issue Adabas direct calls (native commands) directly to an Adabas database for learning and testing and for analyzing problems.

The utility ADACALL is contained in the library SYSADA.

This section covers the following topics:

- Invoking ADACALL
- ADACALL Parameters
- ADACALL Commands and PF Keys
- Adabas OP Command
- User Exit ADAEXIT

Invoking ADACALL

To invoke ADACALL

- Enter the system command SYSADA.
Or enter LOGON SYSADA and then ADACALL.

The ADACALL main screen similar to the example screen below is displayed:

```

15:53:32          ***** NATURAL ADACALL UTILITY *****          2002-04-04
User SAG              - ADABAS Direct Calls -
Mode Char                                Call No. 45
*** Control Block ***          First Byte 30
  Cmd L3          Cmd ID SAG          File 316          Database 10
Resp 0          ISN 382          ISQ 0          ISL 0
FBL 210          RBL 980          SBL 140          VBL 140          IBL 0
COP1          COP2          User Area          Cmd Time 4
Addition1          Addition2 Addition3          Addition4          Addition5
AA]?          227 48
*** Buffer Areas ***
Format AA,AC,AE.

Record 11111003ARTHUR          DENT

Search

Value

ISN
Command ==>
Enter-PF1---PF2---PF3---PF4---PF5---PF6---PF7---PF8---PF9---PF10--PF11--PF12---
      Help Main Exit          Char Hex View Prnt Run Init Canc

```

On the ADACALL main screen, you enter the necessary parameters on the screen and then execute the Adabas command by either choosing PF10 (Run) or entering the ADACALL command EXEC in the command line.

In the example screen above, the Adabas command L3 was executed for a logical read of the employees file.

Except for the control block, which is shown in full, only a part of the buffer is displayed. You can view the buffers in their entirety by using any of the ADACALL direct commands or PF keys listed below.

ADACALL Parameters

The parameters which can be specified are listed below. To obtain a summarized explanation of the parameters, use the ADACALL online help function.

To invoke the online help function

- Place the cursor in the corresponding field and enter a question mark (?) or choose PF1.
(For read-only fields, only PF1 applies.)

For further details, see the Adabas documentation Command Reference and Messages and Codes.

Parameter	Explanation
Mode	Indicates the display mode of the buffer contents: Char Character values. Hex Hexadecimal values. To change modes, see the ADACALL commands CHAR and HEX.
Call No.	Number of commands executed since the start of the session.
First Byte	The first byte of the Adabas control block. Indicates whether 1-byte or 2-byte database IDs (DBID) and file numbers (FNR) are used: H'00' = 1-byte DBID, FNR (file numbers 1 - 255) H'30' = 2-byte DBID, FNR (file numbers greater than 255)
Cmd	Adabas command. Enter and execute the Adabas OP command to specify the parameters described in the relevant section below.
Cmd ID	Command ID.
File	File number. If First Byte is set to H'00': 3-digit file number, "Database" not equal to 0. If First Byte is set to H'30': 5-digit file number.
Database	Database number. Defaults to the DBID of the FUSER file of the current Natural session (see File above). If First Byte is set to H'30', then the database number will be moved to the response code field of the Adabas control block at execution time.
Resp	Response code returned after the command is executed.
ISN	Internal sequence number.
ISQ	ISN quantity.

Parameter	Explanation
ISL	Lowest ISN value for ISN lists.
FBL	Format buffer length in bytes (maximum 210).
RBL	Record buffer length in bytes (maximum 980).
SBL	Search buffer length in bytes (maximum 140).
VBL	Value buffer length in bytes (maximum 140).
IBL	ISN buffer length in bytes (maximum 200).
COP1	Command option 1.
COP2	Command option 2.
User Area	User area for the control block.
Cmd Time	The time taken to execute the command, converted to 1/100th seconds for convenience.
Addition1	Additions 1.
Addition2	Additions 2. If the call was successful, it displays the compressed length of the record being read and the decompressed length of the data requested via the format buffer. If a non-zero response is returned and the error was a result of an invalid format buffer, the field in error and its offset into the format buffer are displayed.
Addition3	Additions 3.
Addition4	Additions 4. If a VSAM file is being read, this field is set to VSAM if initialized.
Addition5	Additions 5.
Format	Format buffer. (The final period is necessary.)
Record	Record buffer.
Search	Search buffer. (The final period is necessary.)
Value	Value buffer.
ISN	ISN buffer.

Adabas OP Command

When you execute the Adabas command OP (Open), ADACALL provides a window where you can specify the following parameters:

- maximum ISNs to be stored in the internal ISN buffer,
- maximum records permitted in hold status,
- maximum CIDs which may be active,
- maximum time permitted for execution of an Sx command.

In the window, enter the relevant information and choose ENTER.

For an explanation of the parameters and valid values, refer to the Adabas Command Reference documentation.

ADACALL Commands and PF Keys

The ADACALL direct commands listed below are provided to change ADACALL parameter settings or to switch between screens by either entering a command in the command line or choosing a corresponding PF key.

In addition to ADACALL commands, from the command line, you can also issue Natural system commands.

Command	PF Key	Function
	PF1	Invoke the help function for ADACALL. If the cursor is positioned on one of the various ADACALL parameters and PF1 is pressed, help information on this parameter is displayed.
	PF2	Return to the ADACALL main screen. Mode is set to CHAR.
BACK	PF5	Page back to the previous buffer when viewing the buffers in their entirety. Valid only after the VIEW command has been applied, which means that the command is not applicable from the ADACALL main screen.
CB		Display the control block buffer entirely; valid in hexadecimal mode only.
CHAR	PF6	Change the current mode to character mode (EBCDIC).
D		Display extended error message text for response code received. When an Adabas response other than 0 (zero) is returned, the corresponding short error message text is displayed in the message line. The extended text can be viewed by issuing this command.
EXEC RUN	PF10	Execute the direct command with the parameters specified.
EXIT STOP Q .	PF3 PF12	Exit. If pressed while on the ADACALL main screen, ADACALL is terminated. If one of the buffer screens is being viewed, the ADACALL main screen is displayed with Mode unchanged.
FB		Display the format buffer in its entirety.
FWD	PF4	Page forward to the next buffer when viewing the buffers in their entirety. Valid only after the VIEW command has been applied, which means that the command is not applicable from the ADACALL main screen.
HEX	PF7	Change the current mode to hexadecimal.
IB		Display the ISN buffer in its entirety.
INIT	PF11	<p>Initialize/reset buffer(s). A window is displayed and one of the following values can be entered for the buffers indicated:</p> <p style="margin-left: 40px;">H Initialize the corresponding buffer(s) with binary zeroes (H'00').</p> <p style="margin-left: 40px;">any character Initialize the corresponding buffer(s) with blanks (H'40'). except H or blank</p> <p style="margin-left: 40px;">blank character Do not initialize the corresponding buffer(s).</p> <p>If you enter INIT ALL, all buffers except the control block are initialized with blanks. Alternatively, the command INIT FB RB SB VB IB (not all buffers need be listed) can be specified and all buffers in the list are initialized with blanks.</p> <p>Note: The ISN buffer is always initialized with binary zeroes.</p>

Command	PF Key	Function
	PF1	Invoke the help function for ADACALL. If the cursor is positioned on one of the various ADACALL parameters and PF1 is pressed, help information on this parameter is displayed.
	PF2	Return to the ADACALL main screen. Mode is set to CHAR.
PRINT	PF9	Generate and display a report on the status of all buffers. The Natural terminal command %H can be used to obtain a hardcopy.
RB		Display the record buffer in its entirety.
RUN		Same as EXEC.
SB		Display the search buffer in its entirety.
VB		Display the value buffer in its entirety.
VIEW	PF8	Display all buffers in their entirety. The first buffer to be displayed is the record buffer. The FWD command can be used to page through the other buffers. If you VIEW the record buffer in hexadecimal mode, the data are displayed on four pages: To page forwards, enter the command FWD or choose PF4. To page backwards, enter the command BACK or choose PF5. To display a specific page, enter a page number from 1 to 4 in the field "Specify next page number". To view buffers individually, enter any of the following commands: FB Format buffer RB Record buffer SB Search buffer VB Value buffer IB ISN buffer CB Control block (default). Valid in hexadecimal mode only: change to HEX before executing VIEW.
VSAM		If VSAM has been defined for the current Natural session, this direct command can be issued to access or update VSAM files. When you issue this command, you are prompted by a window for the VSAM file name. When the command is executed, it is directed to the appropriate VSAM file.

User Exit ADAEXIT

ADACALL allows direct commands to be issued to any database. Therefore, as a means of security, a user exit is supplied. This user exit is called ADAEXIT and is contained in the library SYSADA. You can modify ADAEXIT as required. The Adabas control block is passed as a parameter to ADAEXIT. You can change the source code of the user exit so as to modify the contents of the control block. By simply changing the database ID or file number, or by setting the Command Code to XX, you can prevent database calls from being performed.

SYSBPM Utility - Buffer Pool Management

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

For a general description of the Natural buffer pool, refer to Natural Operations for Mainframes. The buffer pool is defined with the macro NTBPI in the Natural parameter module, or with the corresponding dynamic profile parameter BPI, both of which are also described in Natural Operations for Mainframes.

Some functions of the SYSBPM utility can only be executed if the buffer pool has been initialized with PLUGIN=BP. For local buffer pools, PLUGIN is defined as a Natural profile parameter in the NATPARM parameter module. For global buffer pools, PLUGIN is defined with the CREATE function used to start the buffer pool.

Note:

In the SYSBPM documentation, buffer pool is also referred to as BP.

This documentation covers the following topics:

- **Invoking and Operating SYSBPM** Describes how to invoke the utility and how to select functions by using the SYSBPM Main Menu. Provides information on the maintenance of further buffer pools and the use of SYSBPM in a Sysplex environment.
- **Buffer Pool Statistics** Provides buffer-pool-related, object-independent statistics functions including hash table statistics.
- **BP Cache Statistics** Describes the buffer pool (BP) cache statistics functions including general BP cache statistics, BP cache call statistics, BP cache hash statistics, and individual cache object statistics.
- **Individual Object Statistics** Provides information on the objects currently contained in the buffer pool.
- **Object Directory Information** Displays the full directory of an object currently contained in the buffer pool.
- **Display Object Hexadecimal** Displays an object currently contained in the buffer pool in hexadecimal format.
- **Delete Object from Buffer Pool** Describes how to delete one or more objects from the buffer pool.
- **Blacklist Maintenance** Describes how to maintain a so-called blacklist of Natural objects which are not to be executed and loaded into the buffer pool or, if they are already in the buffer pool, which are to be deleted.
- **Preload List Maintenance** Describes how to maintain a so-called preload list where you can specify the names of Natural objects to be loaded into the buffer pool on its initialization.
- **SYSBPM Direct Commands** Lists the direct commands provided to execute SYSBPM functions.

Invoking and Operating SYSBPM

The SYSBPM utility is menu-driven. You can use a function code, SYSBPM direct commands (see the relevant section) or a PF key to perform a specific function.

This section describes how to invoke the SYSBPM utility and how to select functions by using the SYSBPM Main Menu.

In addition, information is provided on the maintenance of further buffer pools and the use of SYSBPM in a Sysplex environment.

The following topics are covered below:

- Invoking SYSBPM
- SYSBPM Main Menu Functions
- Maintenance of Further Buffer Pools
- SYSBPM in a Sysplex Environment

Invoking SYSBPM

Below is a rough guideline on how to proceed when planning to apply the SYSBPM utility. Before starting the SYSBPM utility, note that only Natural objects but not sources are loaded into the buffer pool.

To invoke the SYSBPM utility

- In the command line, enter the system command SYSBPM.

The SYSBPM Main Menu is displayed:

```

16:31:07          ***** NATURAL SYSBPM UTILITY *****          2002-04-12
BPNAME QA31GBP          - Main Menu -          Type Global NAT
BPPROP OFF          Loc DAEF QA31

          Code  Function
          A    Buffer Pool Statistics
          C    BP Cache Statistics

          S    Individual Object Statistics
          I    Object Directory Information
          O    Display Object Hexadecimal
          D    Delete Object from Buffer Pool

          B    Blacklist Maintenance
          P    Preload List Maintenance

Code .. d    Library ... * _____
              Object .... * _____
              DBID ..... 0_____ FNR .. 0_____

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

```

- On the SYSBPM Main Menu, specify the executed object(s) stored in the buffer pool by choosing either of the two options below:

Complete the input fields as described in SYSBPM Main Menu - Fields and Functions below.

Or, in the command line, enter a direct command as described in SYSBPM Direct Commands.

SYSBPM Main Menu - Fields and Functions

Below are the fields and functions provided on the SYSBPM Main Menu. You invoke the functions by choosing the corresponding Function Code. The functions are described in the relevant sections of the SYSBPM documentation.

Code	Field/Function	Explanation
	BPNAME	Displays the name of the global buffer pool as specified with the profile parameter BPNAME. See the relevant section in the Natural Parameter Reference documentation.
	BPPROP	Displays the setting of the profile parameter BPPROP to control the propagation of changes to an object in a buffer pool. See the relevant section in the Natural Parameter Reference documentation.
	Type	The type of buffer pool, such as Global or Local.
	Loc	Location. Displays the host ID (in the example screen above: DAEF) and the subsystem ID (in the example screen above: QA31).
	Library	The name of the library where the executed object is stored. You can specify a name or use asterisk (*) notation. The default, asterisk (*), selects all libraries.
	Object	The name of the executed object stored in the buffer pool. The default, asterisk (*), selects all objects.
	DBID/FNR	The database ID (DBID) and file number (FNR) of the system file FNAT or FUSER where the executed object is stored and from where it is loaded. If you specify 0 (this is the default) as DBID or FNR, the specified object(s) will be selected regardless of their DBID and FNR. Any value other than 0 represents a valid DBID or FNR specification.
	Code	The code that corresponds to the function desired.
A	Buffer Pool Statistics	This function invokes the BP Statistics Main Menu. From this menu, you can start buffer-pool-related, object-independent statistics functions including hash table statistics.
C	Buffer Pool Cache Statistics	Buffer pool (BP) cache required. This function invokes the BP Cache Statistics Main Menu. From this menu, you can start functions for the BP cache.
S	Individual Object Statistics	This function displays information on objects currently loaded in the Natural buffer pool and the BP cache (if used). Each list item can be accessed individually, and various functions can be performed for each object.

Code	Field/Function	Explanation
	BPNAME	Displays the name of the global buffer pool as specified with the profile parameter BPNAME. See the relevant section in the Natural Parameter Reference documentation.
	BPPROP	Displays the setting of the profile parameter BPPROP to control the propagation of changes to an object in a buffer pool. See the relevant section in the Natural Parameter Reference documentation.
	Type	The type of buffer pool, such as Global or Local.
	Loc	Location. Displays the host ID (in the example screen above: DAEF) and the subsystem ID (in the example screen above: QA31).
	Library	The name of the library where the executed object is stored. You can specify a name or use asterisk (*) notation. The default, asterisk (*), selects all libraries.
	Object	The name of the executed object stored in the buffer pool. The default, asterisk (*), selects all objects.
	DBID/FNR	The database ID (DBID) and file number (FNR) of the system file FNAT or FUSER where the executed object is stored and from where it is loaded. If you specify 0 (this is the default) as DBID or FNR, the specified object(s) will be selected regardless of their DBID and FNR. Any value other than 0 represents a valid DBID or FNR specification.
	Code	The code that corresponds to the function desired.
I	Object Directory Information	This function displays the full directory information of a specified object currently contained in the Natural buffer pool.
O	Display Object Hexadecimal	This function displays in hexadecimal format a specified Natural object that is currently stored in the buffer pool.
D	Delete Object from Buffer Pool	This function is used to delete one or more Natural objects from the buffer pool.
B	Blacklist Maintenance	This function invokes the Blacklist Maintenance menu which is used to maintain a blacklist of Natural objects which are not to be executed.
P	Preload List Maintenance	This function invokes the Preload List Maintenance menu. In a preload list, you can specify the names of Natural objects which are to be loaded into the buffer pool when the buffer pool is initialized.

Maintenance of Further Buffer Pools

With the SYSBPM utility, you can also maintain buffer pools other than the current one.

Three functions are provided for this purpose:

- Display Buffer Pools
- Select Buffer Pool
- Reset Buffer Pool

Display Buffer Pools

When you enter the direct command `DISPLAY BUFFERPOOL`, a window is displayed which contains the following information on your current (global or local) buffer pool and on all further global buffer pools currently available within your Natural system:

- The name of the buffer pool,
- The type of buffer pool (Natural, Sort, DL/I, Editor, Monitor, Instance)
- The current status,
- The name of the preload list (if applicable) and
- The storage address.

Select Buffer Pool

When you enter the direct command `SELECT BUFFERPOOL`, a window is displayed with a list of all buffer pools.

The information displayed is the same as with the function Display Buffer Pools (see above).

In the window, you can select a buffer pool by marking it with any character. Once you have selected a buffer pool from the window, all SYSBPM functions apply to this buffer pool.

Your Natural session itself, however, will continue to run with the original buffer pool.

With the function Reset Buffer Pool (see below), you can switch SYSBPM back to your original buffer pool.

Reset Buffer Pool

If you have used SYSBPM for another buffer pool (see Select Buffer Pool above), you can switch the applicability of SYSBPM back to the buffer pool used by your current Natural session by using the direct command `RESET BUFFERPOOL`.

SYSBPM in a Sysplex Environment

Whenever Natural switches to another operating system image (host), Natural also switches buffer pools. A switch of buffer pools is indicated by a different host ID which is displayed in the Loc field of the SYSBMP screen.

Switching can take place after each terminal I/O, that is, after choosing any function key. After switching, browsing and positioning functions will not be executed (top, bottom, +, -, left, right). Instead, the list starts from the top of the new buffer pool.

If the BPPROP profile parameter (see the relevant section in the Natural Parameter Reference documentation) is set to PLEX or to GPLEX, SYSBPM commands, such as manipulating blacklists, deleting objects or initializing the buffer pool are first executed as usual, and then propagated to other buffer pools available on the same subsystem. If a BP switch caused a function to be aborted or propagated, a corresponding message is issued. There is also a message each time Natural has switched to another host and has changed buffer pools: Attention: BP switched.

SYSBPM - Buffer Pool Statistics

This function invokes the BP (Buffer Pool) Statistics Main Menu which is used to obtain buffer-pool-related, object-independent statistics including hash table statistics.

To invoke Buffer Pools Statistics

- On the SYSBPM Main Menu, enter Function Code **A**.
Or, in the command line, enter `DISPLAY STATISTICS`.

The BP Statistics Main Menu is displayed.

The BP Statistics Main Menu provides the following functions:

- General Buffer Pool Statistics
- Buffer Pool Load/Locate Statistics
- Buffer Pool Fragmentation
- Internal Function Usage
- Buffer Pool Hash Table Statistics

General Buffer Pool Statistics

This function is used to monitor the performance of the buffer pool, and displays various statistics regarding the activity of the buffer pool.

To invoke General Buffer Pool Statistics

- On the BP Statistics Main Menu, enter Function Code **G**.
Or, in the command line, enter `DISPLAY GENERAL`.

The General Buffer Pool Statistics screen is displayed.

The statistics displayed on the General Buffer Pool Statistics screen are snapshots of the buffer pool which are refreshed each time you choose `ENTER`. The following information is displayed:

Field	Explanation
Buffer Pool Address	Shows the storage address of the buffer pool; that is, of the buffer pool control block.
Directory Section	Shows the storage address of the buffer pool directory section (relative to the beginning of the buffer pool). Each object stored in the buffer pool requires a directory entry that contains information on this object. The space for these directory entries is acquired from the buffer pool itself.
Text Record Section	Shows the storage address of the text record section (relative to the beginning of the buffer pool). After the space used by the directory entries has been allocated, the remaining space is divided into blocks called text records (whose size, by default, is 4 KB). A Natural object can occupy one or more text records, depending on its size.
Dataspace Attached	Shows the name of the dataspace attached for the BP cache.

Field	Explanation
Buffer Pool Size (MB)	Shows the size of the whole buffer pool in MB. The buffer pool size can be changed with the NTBPI macro in the parameter module or with the BPI profile parameter (as described in the Natural Reference documentation).
Directory Size	Shows the size of a directory entry in bytes.
Text Record Size	Shows the size of a text record in KB. The text record size can be changed with the NTBPI macro in the parameter module or with the BPI profile parameter (as described in the Natural Reference documentation). The default text record size is set to 4 KB. However, if you use applications that consist of many rather small objects, it is recommended that you reduce it to 2 KB. This reduces the percentage of unused space in the buffer pool, although it can lead to Algorithm 2 (see below) being invoked more frequently.
Initialization	Shows the date and time the buffer pool was initialized.
Last Refresh	Shows the date and time the buffer pool was most recently refreshed, and the ID of the user who performed the refresh.
Text Records - Total	Shows the total number of text records.
Text Records - Used	Shows the number of text records currently used.
Text Records - Used in %	Shows the percentage of text records currently used.
Text Records - Max Used	Shows the maximum number of text records used.
Text Records - Total Size	Shows the total space used by text records, which is the number of active text records times the single text record size. The difference between the total text record size and the total object size shows the amount of unused size in the text record section and can also be an indicator for the system administrator of whether to modify the text record size or not.
Text Records - Avg Usage %	Shows the average usage of text records in percent. This value should not be significantly less than 75%. If the buffer pool is almost full, any value above 75% indicates good usage of the buffer pool. If the usage is significantly less than 75%, the text record size should be reduced.
Space Used %	Shows the actual usage of the buffer pool space in percent. Note: If the buffer pool is almost full (that is, the value in the field Text Records Used is almost 100%), any value above 75% indicates good usage of the buffer pool. If the usage is significantly less than 75%, the text record size should be reduced.
Objects - Loaded	Shows the number of objects currently loaded in the buffer pool.
Objects - Max Loaded	Shows the maximum number of objects loaded in the buffer pool.
Objects - Total Size	Shows the total size in bytes of the objects currently loaded.
Objects - Avg TR Used	Shows the average number of text records used by one object.

Field	Explanation
Objects - SumOfUseCounts	Totals the Use Counts of all objects currently loaded in the buffer pool. The Use Count counts all current users of a given object. If an object is currently not in use, its Use Count returns to 0 (zero).
Objects - AvgLifetimeUsed (min)	Shows the average load time (in minutes) of objects currently loaded in the buffer pool.
Objects - AvgLifetimeReplace (min)	Shows the average load time (in minutes) of objects, which have already been replaced, that is deleted in the buffer pool.

Buffer Pool Load/Locate Statistics

This function provides statistical information on the loading of objects into the buffer pool and the locating of objects in the buffer pool. This information also serves as an indicator of buffer pool performance.

To invoke Buffer Pool Load/Locate Statistics

- On the BP Statistics Main Menu, enter Function Code **L**.
Or, in the command line, enter **DISPLAY LOAD**.

The Buffer Pool Load/Locate Statistics screen is displayed.

The statistics displayed on the Buffer Pool Load/Locate Statistics screen are snapshots of the buffer pool which are refreshed every time you choose **ENTER**. The following information is displayed:

Field	Explanation
Total Locate Calls	Shows the total number of program location calls; that is, the total number of times the Natural buffer pool manager was requested to search the buffer pool for an object. If the location is successful, the object has been loaded from the buffer pool or the BP cache and need not be loaded from a Natural system file thereby saving calls and I/Os.
Total Locate Calls Successful	Shows the total number of successfully performed locate calls as an absolute number.
Total Locate Calls Failed	Shows the total number of locate calls that failed.
Quick Locates	Shows the total number of quick locate calls. Quick location means that the directory address of the last call of the requested program is still available. This is due to the fact that Natural maintains user-specific tables of internal directory entries which contain information on the objects used most recently by each Natural user. When a user invokes an object that has been used before in the Natural session, Natural passes this information to the buffer pool manager, which then bypasses the normal locate procedure. If the last call address cannot be found, a normal locate call is automatically scheduled by the buffer pool manager.
Quick Locates Successful	Shows the number of quick locate calls that have been successfully performed.
Quick Locates Failed	Shows the number of quick locate calls that failed. Failed quick locate calls result in normal locate calls.

Field	Explanation
Normal after Quick	Shows the number of normal locate calls that have been preceded by a quick locate call. For an explanation of normal calls, see the description of Quick Locates above.
Normal after Quick Successful	Shows the number of normal locate calls that have been successful in locating the required Natural object in the buffer pool or the BP cache and have been preceded by a quick locate call.
Normal after Quick Failed	Shows the number of normal locate calls that failed and were preceded by a quick locate call.
Normal Locates	Shows the total number of normal locate calls.
Normal Locates Successful	Shows the number of normal locate calls that were successful in locating the required Natural object in the buffer pool.
Normal Locates Failed	Shows the number of normal locate calls that failed. A failed normal locate call indicates that a program has to be loaded from the database or from the BP cache.
Successful from Cache	Shows the total number of successful locate calls of objects that resided in the BP cache. This information is counted only if the previous locate call (Normal after Quick Failed or Normal Locates Failed) failed. It indicates the number of database loads saved. This means, that, without the BP cache, the object would have to be loaded from the database.
Load Calls (DB)	Shows the total number of load calls made since the buffer pool has been refreshed.
Program Loads	Shows the number of times a Natural object was loaded from a Natural system file into the buffer pool. As several load calls may be necessary to load a single object, this value provides the actual number of program loads made since the most recent buffer pool refresh.
Program Loads - Finished	Shows how many of the program loads above have been successfully executed. For example, a load call could have failed, because at the time of the load, all objects in the buffer pool were currently being executed. When loading a Natural object, the buffer pool manager uses two search algorithms: Algorithm 1 and Algorithm 2 (see below).
Program Loads - Concurrent Loads	Shows the number of object loads that have been performed simultaneously: While an object was loaded by one application with a locate call still in progress, another application requested the same object and the object was loaded more than once.
Algorithm 1	Shows the number of times a storage allocation request satisfied the search criteria of Algorithm 1. The storage allocation request may be triggered either by a load from the database or by a load from the BP cache. Search Algorithm 1 attempts to find a single piece of storage in the buffer pool or a single object that is not currently being executed and can be deleted, in order to obtain the space required to accomplish the load. If an object has to be deleted, Algorithm 1 compares the time stamps of the objects in question and the oldest one will be deleted. The search for space begins at the top of the text record section. If storage is unavailable after Algorithm 1 completes its search, Algorithm 2 is invoked.

Field	Explanation
Algorithm 2	<p>Shows the number of times a storage allocation request satisfied the search criteria of Algorithm 2. The storage allocation request may be triggered either by a load from the database or by a load from the BP cache.</p> <p>Search Algorithm 2 is a more complex search than Algorithm 1. Algorithm 2 attempts to combine two or more entities (free storage and/or objects not being executed) in order to obtain the necessary storage for the load. The ages of the objects, however, are not taken into account.</p> <p>Algorithm 2 begins its search at a point in the text record section where it last left off doing such a search, continues to the bottom of the text record section and, if necessary, wraps around to the top of the text record section to make one final pass from the top to the bottom. If space is still unavailable, Algorithm 2 fails, the object cannot be loaded and a corresponding error message is returned.</p>
Largest Alloc (TR)	Shows the largest single allocation size so far requested, specified in number of text records.
Allocation Failure	Shows the total number of times an object load failed. The reason for a failure is that either all directory entries are in use at the time of the load request or not enough storage is available in the text record section to perform the load.
Allocation Failure - Sizes Failing Last	Shows the number of text records that would have been required by the three most recent storage allocation requests that failed. Storage allocation request failures are a direct result of search Algorithm 2 failing for an object load request.
Total Locate Calls Successful/ Program Loads	<p>These statistics are expressed as a ratio using Total Locate Calls Successful and Program Loads. A value greater than 1 indicates that Natural located more objects in the buffer pool than it loaded from the system file.</p> <p>This ratio serves as a buffer pool efficiency indicator. The larger the number, the better the buffer pool is performing. This is the primary indicator of performance from one buffer pool session to the next.</p>

Buffer Pool Fragmentation

This function provides an overview of the buffer pool fragmentation; that is, an overview of how many different objects occupy how many text records, and how the object locations are spread over the buffer pool.

To invoke Buffer Pool Fragmentation

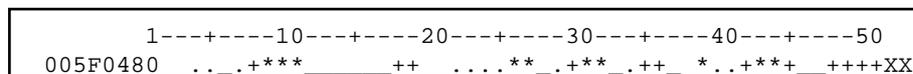
- On the BP Statistics Main Menu, enter Function Code **R**.
Or, in the command line, enter DISPLAY FRAGMENTATION.

The Buffer Pool Fragmentation screen is displayed.

Some of the fields provided on the Buffer Pool Fragmentation are identical with the items explained in General Buffer Pool Statistics above:

- Buffer Pool Size
- Buffer Pool Address
- Text Record Section
- Text Record Size
- Number of Text Records
(same as Text Records - Total)

In addition, the screen displays a diagram which shows how many different individual objects occupy how much text record size. For example:



Each symbol in the diagram represents one text record, and each sequence of equal symbols represents a different individual object occupying one or more text records. The symbols have the following meaning:

_ and .	Objects with a Use Count of 0.
+ and *	Objects with a Use Count greater than 0.
blank character	An unused text record.
XX	The end of the buffer pool, which means that no further text records are available.

In the example above, the buffer pool contains 48 text records. Three of them are not in use; the rest is occupied by 24 different Natural objects, 12 of them with a Use Count of 0, and 12 with a Use Count greater than 0.

Internal Function Usage

This function provides statistical information on the calls to the Natural buffer pool manager.

▶ To invoke Internal Function Usage

- On the BP Statistics Main Menu, enter Function Code **F**.
Or, in the command line, enter DISPLAY FUNCTION.

The Internal Function Usage screen is displayed.

The statistics displayed on the Internal Function Usage screen are snapshots of the buffer pool which are refreshed every time you choose ENTER.

The field Total Calls shows the overall number of all internal calls that have been made to the buffer pool manager.

Internally, the buffer pool manager can be invoked for various different functions. For each function, the number of times it has been invoked is displayed, both as an absolute number and as percentage. In addition, these numbers are represented in a horizontal bar chart.

Buffer Pool Hash Table Statistics

This function displays statistics about hash table slots and collisions per slot. The statistics determine the efficiency of the hash algorithm used.

▶ To invoke Buffer Pool Hash Table Statistics

- On the BP Statistics Main Menu, enter Function Code **H**.
Or, in the command line, enter DISPLAY HASH.

The Hash Table Collisions screen is displayed.

The statistics displayed on the Hash Table Collisions screen are snapshots of the hash table which are taken every time you choose ENTER. The following information is displayed:

Field	Explanation
Total Number of Slots	Shows the total number of hash table slots; that is, the total possible entries that link the object name with the location of the object. The number of slots, that is, the size of the hash table will be calculated internally depending on the number of text records.
Number of Slots Used	Shows the number of slots that have one or more entries.
Number of Slots Free	Shows the number of slots that have no entry.
Max. Collisions per Slot	Shows the maximum number of collisions of all slots. The maximum number of collisions is the longest possible search path for an object.
Collisions	The number of possible collisions. 0 means no collision or one entry. When there are more than 5 collisions, the number of collisions will be specified in ranges (for example, 6 - 10).
Number of Slots	Shows the number of slots grouped by their number of collisions. For example, if the number of collisions is 3, the search algorithm must side step a maximum of 3 times to find an object. In addition, the percentage of these slots related to all slots used is displayed.
Totalled Number of Slots	Shows the same values as Number of Slots, but the values are totalled.

SYSBPM - BP Cache Statistics

This function invokes the BP (Buffer Pool) Cache Statistics Main Menu which is used to obtain statistical information on the BP cache

Note that the BP Cache Statistics function can only be executed if a BP cache has been installed when initializing a global buffer pool (no BP cache support for local buffer pools).

To invoke BP Cache Statistics

- On the SYSBPM Main Menu, enter Function Code **C**.
Or, in the command line, enter `DISPLAY CSTATISTICS`.

The BP Cache Statistics Main Menu is displayed.

From the BP Cache Statistics Main Menu, you can select the following functions:

- General BP Cache Statistics
- BP Cache Call Statistics
- BP Cache Hash Table Statistics

General BP Cache Statistics

This function displays various addresses and statistics regarding the activity of the BP cache.

To invoke General BP Cache Statistics

- On the BP Cache Statistics Main Menu, enter Function Code **G**.
Or, in the command line enter `DISPLAY CGENERAL`.

The General BP Cache Statistics screen is displayed.

The statistics displayed on the General BP Cache Statistics screen are snapshots of the buffer pool, which are refreshed each time you choose ENTER. The following information is displayed:

Field	Explanation
Dataspace - Name	Shows the name of the dataspace where the BP cache resides.
Dataspace - SToken	The term SToken (for Space Token) identifies a dataspace.
Dataspace - ALET	The term ALET (for Address List Entry Token) identifies an index for accessing the dataspace.
Dataspace - Size (MB)	Shows the size of the BP cache in MB.

Field	Explanation
Dataspace - Current State	Shows the status of the BP cache: not initialized locked for init closed free for operation undefined
Dataspace - Initialization	Shows the date and time when the BP cache was initialized.
Internal Buffer Offsets - Header Buffer	Header of the BP cache, contains general BP cache information.
Internal Buffer Offsets - Hash Buffer	Contains the hash table.
Internal Buffer Offsets - Directory Buffer	Shows the storage address of the BP cache directory section (relative to the beginning of the BP cache). Each object stored in the BP cache requires a directory entry that contains information on this object. The space for these directory entries is acquired from the BP cache itself.
Internal Buffer Offsets - Text Buffer	Shows the storage address of the text buffer (relative to the beginning of the BP cache). After allocating the space for all other buffers, the remaining space is divided into text records with a size of 4 KB. A Natural object can occupy one or more text records, depending on its size.
Tot. Text Records	Shows the total number of text records in the BP cache. The number of text records depends on the BP cache size. The text record size for the BP cache is 4KB.
Insert Position	Shows the index number of the text record into which the next object will be inserted. Objects will be inserted into the BP cache when they have to be removed from the buffer pool.
Re-use Cycles	Shows the number of times the BP cache has been completely reused. Every time the BP cache is full, the BP cache manager reuses the BP cache from the start and overwrites the object(s) from there. The objects will remain in the BP cache until the BP cache is used again.
Objects - Max Loaded	Shows the maximum number of objects loaded in the BP cache.
Objects - Loaded	Shows the number of objects currently loaded in the BP cache.

BP Cache Call Statistics

This function provides statistical information on the loading (put), retrieving (get) and deleting of objects into/from the BP cache. This information also serves as an indicator of BP cache performance.

▶ **To invoke BP Cache Call Statistics**

- On the BP Cache Statistics Main Menu, enter Function Code L.
Or, in the command line enter DISPLAY CLOAD.

The BP Cache Call Statistics screen is displayed.

The statistics displayed on the BP Cache Call Statistics screen are snapshots of the buffer pool which are refreshed each time you choose ENTER. The following information is displayed:

Field	Explanation
Search Calls (Total)	Shows the total number of search calls issued from the buffer pool to the BP cache. If an object was found, a search call results in a get call.
Get Calls (from Cache)	Shows the number of get calls the buffer pool issued to load an object from the BP cache into the buffer pool.
Get Calls - Successful	Shows the number of successful get calls issued to the BP cache. A get call is successful if an object requested by the buffer pool was actually loaded from the BP cache into the buffer pool. A get call may be unsuccessful, for example, if an object has been deleted after it was found by the search call.
Put Calls (to Cache)	Shows the total number of put calls issued from the buffer pool to the BP cache.
Put Calls - Successful	An object was really put from the buffer pool to the BP cache.
Put Calls - Obj. already Cached	Shows the number of objects the buffer pool attempted but failed to put into the BP cache since the relevant objects were already available in the BP cache.
Delete Calls	Shows the number of delete calls the buffer pool issued to the BP cache.
Delete Calls - Successful	Shows the number of successful delete calls the buffer pool issued to the BP cache. A delete call is unsuccessful if the object to be deleted is not available in the BP cache.
Get/Put Rate	Shows the ratio of get calls to one put call the buffer pool issued to the BP cache. The value is calculated by dividing the successful get calls by the successful puts calls. The higher the value, the better the BP cache efficiency.
Get/Search Rate	Shows the percentage of successful get calls compared with the total number of search calls the buffer pool issued to the BP cache. The higher the value, the better the cache efficiency.
Initialization	Shows the date and time the BP cache was initialized.
Last Re-use Cycle	Shows the load date and time of the last object that has been overwritten. An object is overwritten in the BP cache when its space has to be reused. This means the load time of the object which has been in the BP cache longest corresponds to the Last Re-use Cycle date and time.
Last Access	Shows the date and time the buffer pool last accessed the BP cache.
Last Put (to Cache)	Shows the date and time the buffer pool last issued a put call to the BP cache.
Last Get (from Cache)	Shows the date and time the buffer pool last issued a get call to the BP cache.
Last Delete	Shows the date and time the buffer pool last issued a delete call to the BP cache.

BP Cache Hash Table Statistics

This function displays statistics about hash table slots and collisions per slot. The statistics determine the efficiency of the hash algorithm used.

To invoke BP Cache Hash Table Statistics

- On the BP Cache Statistics Main Menu, enter Function Code **H**.
Or, in the command line enter DISPLAY CHASH.

The Cache Hash Table Collisions screen is displayed.

The statistics displayed on the Cache Hash Table Collisions screen are snapshots of the hash table which are refreshed every time you choose ENTER. The following information is displayed:

Field	Explanation
Total Number of Slots	Shows the total number of hash table slots; that is, the total possible entries that link the object name with the location of the object. The number of slots, that is, the size of the hash table will be calculated internally depending on the number of text records.
Number of Slots Used	Shows the number of slots that have one or more entries.
Number of Slots Free	Shows the number of slots that have no entry.
Max. Collisions per Slot	Shows the maximum number of collisions of all slots. The maximum number of collisions is the longest possible search path for an object.
Collisions	The number of possible collisions. 0 means no collision or one entry. When there are more than 5 collisions, the number of collisions will be specified in ranges (for example, 6 - 10).
Number of Slots	Shows the number of slots grouped by their number of collisions. For example, if the number of collisions is 3, the search algorithm must side step a maximum of 3 times to find an object. In addition, the percentage of these slots related to all slots used is displayed.
Totaled Number of Slots	Shows the same values as Number of Slots, but the values are totalled.

SYSBPM - Individual Object Statistics

This function invokes the Individual Object Statistics screen where you can obtain statistical data on objects currently located in the buffer pool or the buffer pool (BP) cache (if used). Additionally, you can perform functions for the objects displayed.

▶ To invoke the Individual Object Statistics screen

- On the SYSBPM Main Menu, in the Code field, enter Function Code **S** and specify the object(s): see the valid field input values as described in the section Invoking and Operating SYSBPM.

Or, go directly to the statistics about the buffer pool:

In the command line, enter

DISPLAY INDIVIDUAL *library-name object-name dbid fnr*.

Or, go directly to the statistics about about the BP cache:

In the command line, enter

DISPLAY CINDIVIDUAL *library-name object-name dbid fnr*.

The Individual Object Statistics screen lists all individual objects

1. currently located in the buffer pool (first part of the display) and
2. currently located in the BP cache (second part of the display).

The statistics displayed are snapshots of the contents of the buffer pool which are refreshed every time you choose ENTER.

Note for GDA Objects stored in the Bufferpool:

On the Individual Object Statistics screen two entries may be displayed for a GDA: one entry contains data on the GDA itself and the other entry contains the internal Natural symbol table for this GDA. This may happen if a program has been cataloged that references a GDA (global data area).

Below is information on the statistical data displayed on the screen and the commands available to navigate in the screen, access the objects displayed and manipulate their status:

- Fields and Columns
- Navigation
- Line Commands

Columns

The Individual Object Statistics screen provides the following columns:

Column	Explanation
C	In this column, enter a command to perform a function for the object. The functions are described below.
Library	The library from which the object was loaded.
Object	The name of the object.
DBID	The database ID of the Natural system file from which the object was loaded.
FNR	The file number of the Natural system file from which the object was loaded.

Column	Explanation
Loc	<p>Location of the object :</p> <p>B Buffer pool.</p> <p>B/C Buffer pool and BP cache.</p> <p>C BP cache.</p> <p>C/B BP cache and buffer pool.</p> <p>If B is listed in the first position, the statistical data derive from the buffer pool. If C is listed first, the data derive from the BP cache.</p> <p>Additionally, depending on this positioning, different line commands apply to the fields on the statistics screen (see also Line Commands below).</p>
RLD	<p>Current status of the object in the buffer pool or the BP cache. A BP cache status only refers to object locking and, therefore, is only indicated underneath the L (Locked) of the RLD column.</p> <p>Buffer pool:</p> <p>R Marked as resident. Resident means that the object is not deleted from the buffer pool, not even if the relevant Use Count is set to 0.</p> <p>L Locked while load function is ongoing.</p> <p>D A delete request for the object is pending. It will be deleted from the buffer pool as soon as the value in the Use column is reset to 0.</p> <p>BP cache:</p> <p>L Locked while load function is ongoing.</p> <p>G Locked for get transfer from the buffer pool to the BP cache.</p> <p>D Locked for delete.</p> <p>X Locked.</p>
Use	<p>Buffer pool only.</p> <p>The number of Natural users who are currently executing this object.</p>
Max	<p>Buffer pool only.</p> <p>The maximum number of users who have been executing this object concurrently since it has been loaded into the buffer pool.</p>
Reuse	<p>BP cache only.</p> <p>Indicates how often an object has been loaded (reused) from the BP cache into the buffer pool.</p>
TotalUC	<p>The total number of locate calls of an object since it was loaded into the buffer pool.</p> <p>If a BP cache is used, this value is not lost if the object is removed from the buffer pool and saved to the BP cache. Therefore, this value indicates the number of times this object has been used since it was loaded from the system file.</p> <p>For buffer pool objects, this value is updated regularly. For BP cache objects, this value is only updated after the object was removed from the buffer pool and saved in the BP cache.</p>

Column	Explanation
ObjSize	The size of the object.
Sto	Storage that has to be allocated for the object in the buffer pool or BP cache. For BP cache objects, this value is a multiple of the text record size (4 KB).

Navigation

To scroll in the list of the Individual Object Statistics screen, use the following PF keys and direct commands:

PF Key	Command	Function
PF1		Help. Lists all commands and functions available.
PF4	LAST	Repeats the last command executed.
PF5	CACHE	Only applicable if BP cache data exist. Scrolls to the top of the list with statistical data on BP cache objects.
PF6	--	Scrolls to the top of the list with statistical data on buffer pool objects.
PF7	-	Scrolls up one page.
PF8	+	Scrolls down one page.
PF9	++	Scrolls to the end of the list.
PF10	<	Displays the default screen with the fields truncated underneath the columns Use, Max, Reuse, TotalUC, ObjSize and Sto.
	LEFT	Choose PF11 or enter the line command VA below to display the fields in full length.
PF11	>	Alternative to the line command VA below. Displays the full length of the fields positioned underneath the columns Use, Max, Reuse, TotalUC, ObjSize and Sto.
	RIGHT	Choose PF10 to switch back to the truncated display.

Line Commands

On the Individual Object Statistics screen, in Column C, for each object displayed you can enter any of the line commands listed below:

Command	Function
CL	Buffer pool only. Releases an object marked as resident.
LD	Buffer pool only. Corresponds to the function Object Directory Information.
FO	Buffer pool only. Deletes an object immediately from the buffer pool, regardless of the relevant Use Count.
DE	Marks an object to be deleted from the buffer pool or BP cache. The object is deleted as soon as the relevant Use Count is set to 0 . If issued for a buffer pool object, the object will be deleted from both the buffer pool and the BP cache. If issued for a BP cache object, the object will be deleted from the BP cache only.
RE	Buffer pool only. Marks an object as resident.
OB	Buffer pool only. Corresponds to the function Display Object Hexadecimal.
VA	Displays fields in full length. See also the alternative function key PF11 above.

For each command entered, a confirmation message is displayed for the relevant line overwriting text of existing rows. Possible messages are:

- Failed (in response to any function that has not been executed successfully),
- Deleted (in response to the command DE or FO),
- Released (in response to the command CL) and
- Resident (in response to the command RE).

SYSBPM - Object Directory Information

This function is used to display the full directory of an object currently contained in the buffer pool.

To invoke Object Directory Information

- On the SYSBPM Main Menu, enter Function Code **I** and specify an object:
see the valid field input values as described in the section Invoking and Operating SYSBPM.

Or, in the command line, enter
`DISPLAY DIRECTORY library-name object-name dbid fnr.`

The Object Directory Information screen is displayed.

Below is information on the Object Directory Information screen:

- Fields
 - Functions
-

Fields

The Object Directory Information screen provides the following fields and information on a specified object:

Field	Explanation
Directory of	The type (for example, map) and name of the object.
Loaded from Library	The name of the library from which the object was loaded into the buffer pool.
Loaded - DBID/FNR	The database ID and file number of the system file FNAT or FUSER from which the object was loaded into the buffer pool.
Loaded on	The date and time when the object was loaded into the buffer pool.
Loaded by User	The ID of the user who executed the object.
Last Action on	The date and time when a user last executed the object.
BP Directory at Address	The storage address of the directory of the object in the buffer pool.
Object at Address	The storage address of the object in the buffer pool.
Allocated Size (KB)	The size that has to be allocated to contain the object. It is a multiple of the text record size.
Object Size	The size of the object.
Status (RLD)	<p>R The object is resident in the buffer pool. Resident means that the object is not deleted from the buffer pool, not even if its Use Count is set to 0.</p> <p>L The object is currently being loaded into the buffer pool.</p> <p>D The object is pending a delete request. It is deleted from the buffer pool as soon as its Current Use Count (see below) is set to 0.</p>
Current Use Count	The number of users currently executing the object.
Maximum Use Count	The maximum number of users currently executing the object since it has been loaded into the buffer pool.
Total Use	<p>The total number of users who have executed the object since it was loaded into the buffer pool.</p> <p>If a BP cache is used, this value is not lost if the object is removed from the buffer pool and saved to the BP cache. Therefore, this value indicates the number of times this object has been used since it was loaded from the system file.</p>
Cataloged	The information displayed in the Cataloged section of the Object Directory screen is identical with the information provided with the Natural system command LIST DIRECTORY as described in the relevant section in the Natural Command Reference documentation.



Functions

On the Object Directory Information screen, in the command line, for the object displayed you can enter any of the direct commands listed below:

Command	Function
<u>F</u> DELETE	Deletes an object immediately from the buffer pool, regardless of its Use Count.
<u>R</u> ESIDENT	Marks an object as resident. Resident means that the object is not deleted from the buffer pool, not even if its Use Count is set to 0 .
<u>C</u> LEAR	Releases an object marked as resident.
<u>D</u> ELETE	Marks an object for deletion. See Status D above.
<u>N</u> EXT	Only applies if a range of objects was selected: Displays one object after the other and returns to the screen on which NEXT was entered.

SYSBPM - Display Object Hexadecimal

This function is used to display in hexadecimal format an object currently contained in the buffer pool .

To invoke Display Object Hexadecimal

- On the SYSBPM Main Menu, enter Function Code **O** and specify an object:
see the valid field input values as described in the section Invoking and Operating SYSBPM.

Or, in the command line, enter
DISPLAY OBJECT *library-name object-name dbid fnr*.

The Display Object Hexadecimal screen appears with the object displayed in hexadecimal format.

Navigation

Within the object displayed on the Display Object Hexadecimal screen, you can move to a specific location by entering either an absolute hexadecimal address or a hexadecimal offset relative to your current position.

Choose PF7 to scroll one page backward and PF8 to scroll one page forward.

In addition, on the Display Object Hexadecimal screen, in the direct command line, you can enter any of the following positioning command:

Command	Function
GP	Displays the generated program.
KST	Displays the constant table.
MPT	Displays the multiple-purpose table.
<u>N</u> EXT	Only applies if a range of objects was selected: Displays one object after the other and returns to the screen on which NEXT was entered.

SYSBPM - Delete Object from Buffer Pool

This function is used to delete one object or more from the buffer pool. Unless specified with the DELETE command (see SYSBPM Direct Commands), an object will always be deleted from the buffer pool (BP) cache too.

Objects that have a Current Use Count (see Object Directory Information) of **0** are deleted immediately. Objects of a Current Use Count greater than **0** are marked for deletion and deleted as soon as their Current Use Count is reset to **0**.

To invoke Delete Object from Buffer Pool

- On the SYSBPM Main Menu, enter Function Code **D** and specify the object(s) to be deleted: see the valid field input values as described in Invoking and Operating SYSBPM.

Or, in the command line, enter
DELETE *library-name object-name dbid fnr*.

SYSBPM - Blacklist Maintenance

This function is used to maintain a so-called blacklist of Natural objects which are not to be executed and loaded into the buffer pool or, if they already are in the buffer pool, are to be deleted. If the BP cache is enabled, the objects will also be deleted from the BP cache. The blacklist always applies to the buffer pool currently active.

As described below, on the blacklist you can maintain individual objects and object sets which contain several objects. In an object set, you specify the objects not to be executed and add a single set (instead of multiple individual objects) to the blacklist. You can also combine both: maintain objects individually or by sets. An object set is stored as a Natural object of the type Text.

For details on the blacklist, see the relevant section in Natural Buffer Pool in the Natural Operations for Mainframes documentation.

To invoke Blacklist Maintenance

- On the SYSBPM Main Menu, enter Function Code **B**.
Or, in the command line, enter BLACKLIST.

The Blacklist Maintenance menu is displayed.

The functions provided on the Blacklist Maintenance menu are listed and explained below:

- Maintain Blacklist
- List Object Sets
- Edit Object Set
- Add Object Set to Blacklist
- Delete Object Set from Blacklist

Also below is information on:

- Delete Object Set Text Member
 - BPMBLBAT - Blacklist Maintenance in Batch Mode.
-

Maintain Blacklist

This function invokes the Maintain Blacklist screen where you can display and maintain all objects currently available on the blacklist.

To invoke the Maintain Blacklist screen

- On the Blacklist Maintenance menu, enter Function Code **M**.
Or, in the command line, enter DISPLAY BLACKLIST.

The Maintain Blacklist screen appears and displays the current blacklist.
Choose PF7 to scroll one page backward and PF8 to scroll one page forward.

Depending on the mode set when calling the Maintain Blacklist function earlier during a SYSBPM session, the Maintain Blacklist screen appears in Display Mode (default when initializing SYSBPM) or Add Mode. Use PF9 to switch from one mode to the other.

- Adding Objects
- Modifying Objects
- Deleting Objects

Adding Objects

▶ To add objects to the blacklist

1. On the Blacklist Maintenance menu, enter Function Code **M**.
Or, in the command line, enter DISPLAY BLACKLIST.

The Maintain Blacklist screen is displayed.

2. If required, choose PF9 to switch to Add Mode.
A screen with empty input fields appears.
3. In the relevant input fields, enter the name of the library where the objects are stored, the names of the objects and the corresponding database IDs (DBID) and file numbers (FNR).

If DBID and FNR are left blank, they will be taken from the current system file FUSER or FNAT in libraries whose names start with SYS.

If you want to clear the Add Mode screen, in the direct command line, enter CLE or CLEAR.

4. Choose PF5 to confirm the addition
Or, in the command line, enter UP or UPDATE.

A corresponding message appears.

Modifying Objects

▶ To modify objects on the blacklist

1. On the Blacklist Maintenance menu, enter Function Code **M**.

Or, in the command line, enter DISPLAY BLACKLIST.

The Maintain Blacklist screen is displayed.

2. If required, choose PF9 to switch to Display Mode and obtain the list of all objects currently on the blacklist.
3. In the relevant input field(s), replace the existing entries with new values.
4. Choose PF5 to confirm the modification.
Or, in the command line, enter UP or UPDATE.

A corresponding message appears.

Deleting Objects

▶ To delete individual objects from the blacklist

1. On the Blacklist Maintenance menu, enter Function Code **M**.
Or, in the command line, enter DISPLAY BLACKLIST.

The Maintain Blacklist screen is displayed.

2. If required, choose PF9 to switch to Display Mode and obtain a list of all objects currently on the blacklist.
3. In Column **C**, next to the object(s) desired, enter Line Command **DE**.
4. Choose ENTER to confirm the deletion.

A corresponding message appears.

▶ To delete all objects from the blacklist

1. On the Blacklist Maintenance menu, enter Function Code **M**.
Or, in the command line, enter `DISPLAY BLACKLIST`.

The Maintain Blacklist screen is displayed.

2. Choose PF2.
The Confirm Delete window is displayed:
 - To execute the deletion, enter Y (Yes).
 - To cancel the deletion:
Choose PF3 without entering anything in the window,
Or enter or confirm N (No) which is the default.
3. Choose ENTER to confirm the action. A corresponding message appears.

List Object Sets

This function invokes the List Object Sets screen which displays a list of all existing object sets.

To invoke the List Object Sets screen

- On the Blacklist Maintenance menu, enter Function Code **L**, a library name and an object set name. Asterisk (*) notation is also allowed for an object set name.

Or, in the command line, enter `LIST SET library-name set-name`.
Asterisk (*) notation is also allowed for *set-name*.

The List Object Sets screen appears and displays the specified set(s).

You can manipulate an object set from the List Object Sets screen by using any of the line commands provided to modify a set and add it to or delete it from the blacklist. For a list of possible commands, enter a question mark (?) in any of the leftmost screen columns which contain the prefix information.

Edit Object Set

This function invokes the Edit Object Set screen where you can create a new object set, add objects to an existing set or modify them, or delete objects from a set.

The editing functions provided on the Edit Object Set screen are a subset of the functions provided by the Software AG Editor as described in the relevant documentation. To invoke the Help window with a list of the commands available, in the command line, enter a question mark (?). Choose PF7 to scroll backward and PF8 to scroll forward in the window.

For a list of the line commands available, in any of the leftmost columns (prefix information), type in a question mark (?).

Below is information on:

- Creating Object Sets
- Modifying Object Sets

Creating Object Sets

To create an object set

- On the Blacklist Maintenance menu:
 - Enter Function Code **E**.
 - Enter the name of a library.
 - Do **not** enter the name of an object set but clear the contents (if any) of the corresponding field.

The Edit Object Set screen is displayed.

- In the relevant input fields, enter the name of the library where the objects are stored, the names of the objects and the corresponding database IDs (DBID) and file numbers (FNR).
If DBID and FNR are left blank, they will be taken from the current system file FUSER or FNAT in libraries whose names start with SYS.
- In the command line, enter *SA set-name* to save the object set as Natural Text member.

Modifying Object Sets

Below are the functions provided to add an object to an object set, to modify existing objects or to delete them from the set. Note that any of these object set modifications will **not** update the current blacklist.

To add a new object to an object set

- On the Blacklist Maintenance menu, enter Function Code **E**, a library name and an object set name.

Or, on the List Object Sets screen, in the leftmost column, next to the object set desired, enter Line Command **E**.

Or, in the command line, enter
EDIT SET library-name set-name.

The Edit Object Set screen appears and displays the specified object.

- Complete the input fields by entering the name of the library where the objects are stored, the names of the objects and the corresponding database IDs (DBID) and file numbers (FNR).
If DBID and FNR are left blank, they will be taken from the current system file FUSER or FNAT in libraries whose names start with SYS.
- In the command line, enter *SA* to save the modification.

To modify an object of an object set

- On the Blacklist Maintenance menu, enter Function Code **E**, a library name and an object set name.

Or, on the List Object Sets screen, in the leftmost column, next to the object set(s) desired, enter the line command **E**.

Or, in the command line, enter
EDIT SET library-name set-name.

The Edit Object Set screen appears and displays the specified object set.

- In the relevant input field(s), replace the existing entries with new values.
- In the command line, enter *SA* to save the modification.

To delete an object from an object set

- On the Blacklist Maintenance menu, enter Function Code **E**, a library name and an object set name.

Or, on the List Object Sets screen, in the leftmost column, next to the object set(s) desired, enter Line Command **E**.

Or, in the command line, enter
EDIT SET *library-name set-name*.

The Edit Object Set screen appears and displays the specified object set.

- In the leftmost column, next to the object desired, enter Line Command **D** and choose ENTER.
- In the command line, enter SA to save the modification.

Add Object Set to Blacklist

This function is used to add all objects of an object set to the blacklist.

To add an object set to the blacklist

- On the Blacklist Maintenance menu, enter Function Code **A**, a library name and an object set name.

Or, on the List Object Sets screen, in the leftmost column, next to the object set(s) desired, enter Line Command **AC**.

Or, on the Edit Object Set screen, in the command line, enter **AC**.

Or, in the command line, enter ADD SET *library-name set-name*.

A message appears confirming that the object set was added to the blacklist.

Note:

The command **AC** denotes ACTIVATE which is the equivalent of Add Object Set to Blacklist.

Delete Object Set from Blacklist

This function is used to delete all objects of an object set from the blacklist. Note that the Delete Object Set function will **not** delete the object set as a Natural Text member. The objects of the object set can be added to the blacklist again at any time, as described above. See also To delete an object set Text member below.

To delete an object set from the blacklist

- On the Blacklist Maintenance menu, enter Function Code **D**, a library name and an object set name.

Or, on the List Object Sets screen, in the leftmost column, next to the object set(s) desired, enter Line Command **DA**.

Or, on the Edit Object Set screen, in the command line, enter **DA**.

Or, in the command line, enter
DELETE SET *library-name set-name*.

A message appears confirming that the object set was deleted from the blacklist.

Note:

The command **DA** denotes DEACTIVATE which is the equivalent of Delete Object Set from Blacklist.

Delete Object Set Text Member

To delete an object set Text member

- On the Blacklist Maintenance menu, enter Function Code **L**, a library name and an object set name.

Or, in the command line, enter LIST SET *library-name list-name*.

The List Object Sets screen is displayed.

- In the leftmost column, next to the object desired, enter Line Command **D** and choose ENTER. The Delete window appears.
- Confirm the deletion by entering the name of the object set. A corresponding confirmation message appears.

Note that the deletion of an object set Text member **will not** update the current blacklist.

BPMBLBAT - Blacklist Maintenance in Batch Mode

Online, you can lock individual objects against being executed by using the Blacklist Maintenance functions as described above.

In batch mode, you do this by using a Natural batch job that uses the program BPMBLBAT in the library SYSBPM.

The program BPMBLBAT is used as follows:

- Start a Natural batch job in the usual way.
- The CMSYNIN file instructs the Natural nucleus to log on to SYSBPM and to execute the program BPMBLBAT.
- The next command from CMSYNIN is the FIN command.
- BPMBLBAT reads the input from the CMOBJIN file, where the first card must be in either of the following two formats:

Format 1: **FUNC=LOCK, BPNAME=name, LIB=name, DBID=nnn, FNR=nnn**

Format 2: **FUNC=RLS, BPNAME=name**

Format 1 causes objects to be included in the blacklist of the specified buffer pool (BPNAME=name). The following cards must contain in Positions 1 to 8 the name of the object to be locked. For each card, an entry is added to the blacklist with the specified object name and the corresponding library name, DBID and FNR. The last card must contain a period (.) to indicate the end of the input.

Format 2 causes the blacklist to be deleted, which means that no objects are locked in the specified buffer pool.

Example 1:

This JCL is an example of how to lock Programs A, B and C in Buffer Pool V23GBP:

```
//SAGBAT      JOB      ,T.TEST,CLASS=K,MSGCLASS=X,REGION=2048K
//*
//NATURAL     EXEC     PGM=NAT220OBT,PARM='IM=D,OBJIN=Y'
//STEPLIB    DD      DSN=OPS.SYSF.TESTNAT.LOAD,DISP=SHR
//           DD      DSN=OPS.SYSF.V5.ADALOD,DISP=SHR
//DDCARD     DD      *
ADARUN PROGRAM=USER,SVC=249,DATABASE=10,MODE=MULTI
//SYSOUT     DD      SYSOUT=X
//SYSUDUMP   DD      SYSOUT=X
//CMSYNIN    DD      *
LOGON SYSBPM
BPMBLBAT
FIN
```

```
//CMBJIN      DD      *
FUNC=LOCK ,BPNAME=V23GBP , LIB=SAGTEST , DBID=10 , FNR=32
A
B
C
.
//CMPRINT    DD      SYSOUT=X
//
```

Example 2:

This JCL is an example of how to set to **0** (zero) the number of locked entries in Buffer Pool V23GBP:

```
//SAGBAT      JOB      ,T.TEST ,CLASS=K ,MSGCLASS=X ,REGION=2048K
//*
//NATURAL     EXEC     PGM=NAT220OBT ,PARM=' IM=D ,OBJIN=Y '
//STEPLIB     DD      DSN=OPS.SYSF.TESTNAT.LOAD ,DISP=SHR
//           DD      DSN=OPS.SYSF.V5.ADALOD ,DISP=SHR
//DDCARD      DD      *
ADARUN PROGRAM=USER ,SVC=249 ,DATABASE=10 ,MODE=MULTI
//SYSOUT      DD      SYSOUT=X
//SYSUDUMP    DD      SYSOUT=X
//CMSYNIN     DD      *
LOGON SYSBPM
BPMBLBAT
FIN
//CMBJIN      DD      *
FUNC=RLS ,BPNAME=V23GBP
//CMPRINT    DD      SYSOUT=X
//
```

SYSBPM - Preload List Maintenance

This function is used to maintain so-called preload lists. In a preload list, you can specify the names of Natural objects which are to be loaded into the buffer pool when the buffer pool is initialized.

The preload lists themselves are stored as Natural objects of the type Text in the library SYSBPM.

For further details on the preload list, see the relevant section in Natural Buffer Pool in the Natural Operations for Mainframes documentation.

To invoke Preload List Maintenance

- On the SYSBPM Main Menu, enter Function Code **P**.
Or, in the command line, enter PRELOADLIST.

The Preload List Maintenance menu is displayed.

The functions provided on the Preload List Maintenance menu are listed and explained below:

- List Preload Lists
- Edit Preload List
- Generate Preload List from Buffer Pool

Also below is information on:

- Delete Preload List
-

List Preload Lists

This function invokes the List Preload Lists screen which displays a list of all existing preload lists.

To invoke List Preload Lists

- On the Preload List Maintenance menu, enter Function Code **L** and the name of a preload list.
Asterisk (*) notation is also allowed for a preload list name.

Or, in the command line, enter LIST PRELOADLIST *list-name*.
Asterisk (*) notation is also allowed for *list-name*.

- The List Preload Lists screen is displayed.

For a list of possible line commands, enter a question mark (?) in any of the leftmost screen columns which contain the prefix information.

Further commands can be entered in the command line. To invoke the Help window with a list of commands available, enter a question mark (?) in the command line. Choose PF7 to scroll backward and PF8 to scroll forward in the window.

Edit Preload List

This function invokes the Edit Preload List screen where you can create a new preload list, add objects to an existing list or delete objects from it.

Attention:

The editing functions provided on the Edit Preload List screen are a subset of the functions provided by the Software AG Editor (see the relevant documentation). Therefore, before you start a Natural session to edit a preload list, set the Natural profile parameter EDPSIZE to a value greater than zero (see also Profile Parameters in the Natural Parameter Reference documentation). We recommended that you set EDPSIZE to a minimum of 100.

To invoke the Help window with a list of the commands available, in the command line, enter a question mark (?). Choose PF7 to scroll backward and PF8 to scroll forward in the window.

For a list of the line commands available, in any of the leftmost columns (prefix information), enter a question mark (?).

Below is information on:

- Creating Preload Lists
- Modifying Preload Lists

Creating Preload Lists

To create a preload list

- On the Preload List Maintenance menu:
 - Enter Function Code **E**.
 - Clear the contents of the field Preload List Name, that is, do **not** enter the name of a preload list.
 - In the fields Generation Library and Generation Objects, leave the default asterisk (*).
 The Edit Preload List screen is displayed.
- In the relevant input fields, enter the name of the library where the objects are stored, the names of the objects and the corresponding database IDs (DBID) and file numbers (FNR).
If DBID and FNR are left blank, they will be taken from the current system file FUSER or FNAT in libraries whose names start with SYS.
The resident flag will be set to **Y** (Yes) in Column **R** on the editing screen if no value is entered. Resident means that the object is not deleted from the buffer pool, not even if its Use Count is set to **0**.
- In the command line, enter `SA set-name` to save the object set as Natural Text member in the library SYSBPM.

See also Generate Preload List from Buffer Pool below.

Modifying Preload Lists

To add a new object to a preload list

- On the Preload List Maintenance menu, enter Function Code **E** and the name of a preload list.

Or, on the List Preload Lists screen, in the leftmost column, next to the preload list desired, enter Line Command **E**.

Or, in the command line, enter `EDIT PRELOADLIST list-name`.

The Edit Preload List screen appears and displays the specified preload list.
- Complete the input fields by entering the name of the library where the objects are stored, the names of the objects and the corresponding database IDs (DBID) and file numbers (FNR).
If DBID and FNR are left blank, they will be taken from the current system file FUSER or FNAT in libraries whose names start with SYS.
- In the command line, enter `SA` to save the modification.

▶ **To modify an object of a preload list**

- On the Preload List Maintenance menu, enter Function Code **E**, a library name and the name of a preload list.

Or, on the List Preload Lists screen, in the leftmost column, next to the object set desired, enter Line Command **E**.

Or, in the command line, enter `EDIT PRELOADLIST list-name`.

The Edit Preload List screen appears and displays the preload list specified.

- In the relevant input field(s), replace the existing entries with new values.
- In the command line, enter **SA** to save the modification.

▶ **To delete an object from a preload list**

- On the Preload List Maintenance menu, enter Function Code **E**, a library name and the name of a preload list.

Or, on the List Preload Lists screen, in the leftmost column, next to the object set(s) desired, enter Line Command **E**.

Or, in the command line, enter `EDIT PRELOADLIST list-name`.

The Edit Preload List screen appears and displays the preload list specified.

- In the leftmost column, next to the object desired, enter Line Command **D** and choose **ENTER**.
- In the command line, enter **SA** to save the modification.

Generate Preload List from Buffer Pool

This function is used to generate a new preload list using the objects contained in the buffer pool; that is, from the objects that are currently in the buffer pool, you can select those you wish to be included in the preload list.

▶ **To generate a preload list use either of the two options below**

1. On the Preload List Maintenance menu, enter Function Code **G**, the name of a preload list, and, in the fields Generation Library and Generation Objects, specify the objects to be included in the list:
 - To include all objects that are currently in the buffer pool, enter an asterisk (*) in each of the two fields.
 - To include specified objects in the buffer pool, enter any combination of asterisks (*), full names and names with asterisk (*) notation for both fields.

Example:

To include in the preload list all objects which have been loaded from a specific library, enter the full name of the library in the field Generation Library and an asterisk (*) in the field Generation Objects.

2. Or, in the command line, enter `GENERATE PRELOADLIST list-name` or `GENERATE PRELOADLIST list-name gen-library` (See also the field and parameter explanations above).

A message appears confirming that the preload list was generated from the buffer pool.

All preload list objects will be generated as resident (entry **Y** in Column **R**) by default. Choose manually, which objects you want to remove from the list.

Objects from the library SYSBPM will not be included in the generated preload list as it can be assumed that these are objects which were only loaded into the buffer pool in order to execute this function.

Delete Preload List

To delete a preload list

- On the Preload List Maintenance menu, enter Function Code **L** and the name of a preload list.

Or, in the command line, enter `LIST PRELOADLIST list-name`.

The List Preload Lists screen is displayed.

- In the leftmost column, next to the object desired, enter Line Command **D** and choose ENTER. The Delete window appears.
- Confirm the deletion by entering the name of the preload list. A corresponding confirmation message appears.

SYSBPM Direct Commands

The SYSBPM utility provides commands to directly execute SYSBPM functions or navigate through screens.

You enter a direct command in the command line on any SYSBPM screen. An underlined portion of a keyword represents an acceptable abbreviation. Letters in italics are used to represent variable information. You must supply a valid value when specifying this term.

The following direct commands are available in SYSBPM:

Command	Parameters	Function
<u>ADD</u> <u>BL</u> ACKLIST	none	Invokes the Maintain Blacklist screen.
<u>ADD</u> <u>SET</u>	<i>library-name set-name</i>	Adds all objects of a specified object set to a blacklist as described in Add Object Set to Blacklist.
<u>BL</u> ACKLIST	none	Invokes the Blacklist Maintenance menu.
<u>BO</u> TTOM	none	Scrolls to the end of a list.
<u>CA</u> NCEL	none	Same as EXIT.
<u>CHE</u> CK <u>HA</u> SH <u>CHE</u> CK HT	none	PLUGIN=BP required. Checks the BP hash table for consistency and returns the number of inconsistencies found. See also REBUILD HASH.
<u>CL</u> OSE BPC	none	BP cache required. Invokes the function Close BP Cache. The buffer pool runs without BP cache afterwards. You can restart the BP cache using the INITIALIZE BPC command.
<u>CL</u> OSE <u>HA</u> SH <u>CL</u> OSE HT	none	PLUGIN=BP required. Invokes the function Close BP Hash Table which causes the buffer pool to run without using the hash table algorithm and without recording hash table statistics. To reactivate the hash table algorithm, use REBUILD HASH or initialize the buffer pool.
<u>DE</u> LETE	none	Deletes all objects from the buffer pool (BP) and the BP cache (BPC). If entered on the Object Directory Information screen: see DELETE under Functions in the relevant section.
<u>DE</u> LETE	<i>library-name object-name dbid fnr</i>	Deletes the specified object(s) from the buffer pool (BP) and the BP cache (BPC) as described in Delete Object from Buffer Pool.
<u>DE</u> LETE ALL	none	Deletes all objects from the blacklist as described in Delete Object from Blacklist.

Command	Parameters	Function
<u>DELETE BUFFERPOOL</u> <u>DELETE BP</u>	none <i>library-name</i> <i>object-name dbid fnr</i>	Deletes all objects from the buffer pool (BP) only. Deletes the specified object(s) from the buffer pool (BP) only.
<u>DELETE BPC</u>	none <i>library-name</i> <i>object-name dbid fnr</i>	BP cache required. Deletes all objects from the BP cache (BPC) only. BP cache required. Deletes the specified object(s) from the BP cache (BPC) only.
<u>DELETE BLACKLIST</u>	none	Invokes the Maintain Blacklist screen where you can delete blacklist entries.
<u>DELETE SET</u>	<i>library-name set-name</i>	Deletes all objects of a specified object set from the blacklist as described in Delete Object Set from Blacklist.
<u>DISPLAY ALL</u>	none	Same as DISPLAY INDIVIDUAL.
<u>DISPLAY BUFFERPOOL</u> <u>DISPLAY BP</u>	none	See Display Buffer Pools in Invoking and Operating SYSBPM.
<u>DISPLAY BLACKLIST</u>	none	Invokes the Maintain Blacklist screen.
<u>DISPLAY CGENERAL</u>	none	BP cache required. Invokes the General BP Cache Statistics screen.
<u>DISPLAY CHASH</u>	none	BP cache required. Invokes the function BP Cache Hash Table Statistics and displays the Cache Hash Table Collisions screen.
<u>DISPLAY CINDIVIDUAL</u>	<i>library-name</i> <i>object-name dbid fnr</i>	BP cache required. Invokes the Individual Cache Object Statistics screen. In contrast to the command DISPLAY INDIVIDUAL (see below), this command generates a statistics report that displays data about BP cache objects at the beginning of the list.
<u>DISPLAY CLOAD</u>	none	BP cache required. Invokes the BP Cache Call Statistics screen.
<u>DISPLAY CSTATISTICS</u>	none	BP cache required. Invokes the BP Cache Statistics Main Menu.
<u>DISPLAY DIRECTORY</u>	<i>library-name</i> <i>object-name dbid fnr</i>	Invokes the Object Directory Information screen.
<u>DISPLAY FRAGMENTATION</u>	none	Invokes the Buffer Pool Fragmentation screen.
<u>DISPLAY FUNCTION</u>	none	Invokes the Internal Function Usage screen.

Command	Parameters	Function
<u>DISPLAY</u> <u>GENERAL</u>	none	Invokes the General Buffer Pool Statistics screen.
<u>DISPLAY</u> <u>HASH</u> <u>DISPLAY</u> <u>HT</u>	none	PLUGIN=BP required. Invokes the function Buffer Pool Hash Table Statistics and displays the Hash Table Collisions screen.
<u>DISPLAY</u> <u>INDIVIDUAL</u>	<i>library-name</i> <i>object-name dbid fnr</i>	Invokes the Individual Object Statistics screen. In contrast to the command <u>DISPLAY</u> <u>CINDIVIDUAL</u> (see above), this command generates a statistics report that displays data about buffer pool objects at the beginning of the list.
<u>DISPLAY</u> <u>LOAD</u>	none	Invokes the Buffer Pool Load/Locate Statistics screen.
<u>DISPLAY</u> <u>OBJECT</u>	<i>library-name</i> <i>object-name dbid fnr</i>	Invokes the Display Object Hexadecimal screen.
<u>DISPLAY</u> <u>STATISTICS</u>	none	Invokes the BP Statistics Main Menu.
<u>EDIT</u> <u>PRELOADLIST</u>	<i>list-name</i>	Invokes the Edit Preload List screen.
<u>EDIT</u> <u>SET</u>	<i>library-name set-name</i>	Invokes the Edit Object Set screen as described in Blacklist Maintenance.
<u>EXIT</u>	none	Leaves the current function/screen and displays the previous screen.
<u>FLIP</u>	none	Switches the PF-key line.
<u>GENERATE</u> <u>PRELOADLIST</u>	<i>list-name gen-library</i>	Invokes the function Generate Preload List from Buffer Pool.
<u>INITIALIZE</u>	none, 1, 2, 4, 8, 12, 16	Reinitializes the buffer pool and the BP cache. If no text record size is specified, the current text record size will be taken.
<u>INITIALIZE</u> <u>BP</u>	none, 1, 2, 4, 8, 12, 16	Reinitializes the buffer pool only. If no text record size is specified, the current text record size will be taken.
<u>INITIALIZE</u> <u>BPC</u>	none	BP cache required. Reinitializes the BP cache only. To avoid program abends of other users, it is recommended to close the BP cache before initializing it. The text record size of the BP cache is fixed (4 KB).
<u>INITIALIZE</u> <u>OLD</u>	none, 1, 2, 4	Reinitializes the buffer pool with the old format. If no text record size is specified, the current text record size will be taken. This enables sessions with the parameter setting <code>PLUGIN=NOBP</code> to access this buffer pool (fallback).
<u>LAST</u>	none	Displays the SYSBPM direct command entered most recently.
<u>LIST</u> <u>PRELOADLIST</u>	<i>list-name</i>	Invokes the List Preload Lists screen for the specified object.

Command	Parameters	Function
<u>L</u> IST <u>S</u> ET	<i>library-name set-name</i>	Invokes the List Object Sets screen for the specified library or object as described in Blacklist Maintenance. Asterisk (*) is also allowed for <i>set-name</i> .
MENU	none	Invokes the SYSBPM Main Menu as described in Invoking and Operating SYSBPM.
<u>P</u> RELOADLIST	none	Invokes the Preload List Maintenance menu.
QUIT	none	Same as EXIT.
<u>R</u> EBUILD <u>H</u> ASH <u>R</u> EBUILD <u>H</u> T	none	PLUGIN=BP required. This function is used to rebuild hash tables if inconsistencies are found with CHECK HASH. REBUILD HASH deletes the current hash table and rebuilds a new hash table from the current buffer pool contents.
<u>R</u> ESET <u>B</u> UFFERPOOL <u>R</u> ESET <u>B</u> P	none	See Reset Buffer Pool in Invoking and Operating SYSBPM.
<u>S</u> ELLECT <u>B</u> UFFERPOOL <u>S</u> ELLECT <u>B</u> P	none	See Select Buffer Pool in Invoking and Operating SYSBPM.
STOP	none	Leaves the SYSBPM utility.
<u>T</u> OP	none	Scrolls to the beginning of a list.
+	none	Scrolls one page down in a list.
-	none	Scrolls one page up in a list.

SYSEDT Utility - Editor Buffer Pool Services

The Editor Buffer Pool Services utility SYSEDT is intended for Natural administrators and used to:

- Display parameters and runtime information of the editor buffer pool,
- Modify parameters,
- Delete logical work and recovery files.

This section covers the following topics:

- Defining a Natural Security Library Profile
 - Invoking and Operating SYSEDT
 - General Information
 - Generation Parameters
 - Users
 - Logical Files
 - Recovery Files
 - System Administration Facilities
-

Defining a Natural Security Library Profile

If you have Natural Security installed, you must create a library security profile for the SYSEDT utility.

For details, see Library Maintenance as described in the Natural Security documentation.

Invoking and Operating SYSEDT

Below is information on:

- Invoking the SYSEDT Utility
- Invoking a SYSEDT Function
- Using Direct Commands Help

Invoking the SYSEDT Utility

To invoke the SYSEDT utility

- In the direct command line, enter SYSEDT.

The SYSEDT Main Menu is displayed with the following functions:

- General Information
- Generation Parameters
- Users
- Logical Files
- Recovery Files
- System Administration Facilities

The SYSEDT functions are explained in the remainder of this section.

Invoking a SYSEDT Function

To invoke a SYSEDT function

- On the SYSEDT Main Menu, enter the corresponding function code.
Or, on the SYSEDT Main Menu, press the appropriate PF key.
Or, skip the SYSEDT Main Menu and access the desired function direct by entering the direct command SYSEDT followed by the corresponding function code in the NEXT line.

The individual functions are described in the following section.

Using Direct Commands Help

If you enter a question mark (?) in the command line, all direct commands available within the SYSEDT utility are displayed in alphabetical order.

General Information

To invoke the General Information function

- On the SYSEDT Main Menu, enter Function Code **G**.
Or, on the SYSEDT Main Menu, press PF10 (GInfo).

The General Information screen is displayed which provides an overview of the current status of the editor buffer pool:

Item	Shows
Usage Statistics	The currently available total number, the currently used number, and the currently used percentage of the available number of the items that follow.
Buffer Pool Blocks	The number of blocks in the editor buffer pool.
Work File Records	The number of records in the editor work file.
Control	The number of control records, which is always one.
Work	The number of work records.
Recovery Records	The number of recovery records.
Logical Files	The number of logical files.
Requests	The total number of read and write requests, the number of read and write requests for buffer pool blocks (Pool column), and the number of read and write requests for work or recovery files (File column). The Copy column shows read requests, which (in contrast to locked read requests) result in the deletion of the corresponding buffer pool block.
Read Work	The number of read requests for logical file records. A logical file record can be found in the buffer pool (Pool column) or on the work file (File column). It can be read by a locked or by a copy request: locked means that the record is kept in the buffer pool for some time; copy means that it is deleted from the buffer pool after having been read.
Write Work	The number of write requests for logical file records. A record can be either written to the buffer pool (Pool) or moved to the work file (File) if there are no free blocks available.
Read Recovery	The number of read requests for recovery records in the editor work file.
Write Recovery	The number of write requests for recovery records in the editor work file.
Timeout Values	Shows items with timeout values specified in seconds. These timeout values can be modified after pressing PF5 (Updat) and dynamically set after pressing PF5 (Save) again. The modified values are not kept during a restart of the buffer pool. The values from the work file control record are used instead.
Logical Files	The time after which a logical file is deleted if it has not been accessed during this time.
Files Delete Check	The time after which all logical files are checked periodically whether they can be deleted.
Changed Blocks	The time after which blocks that have been modified can be freed by writing them to the work file.
Unchanged Blocks	The time after which blocks that have not been modified can be freed by writing them to the work file.
Locked Blocks	The time after which blocks that have been read with locked can be freed by writing them to the work file.

Generation Parameters

 **To invoke the Generation Parameters function**

- On the SYSEDT Main Menu, enter Function Code **P**.
Or, on the SYSEDT Main Menu, press PF11 (Parms).

The Generation Parameters screen appears.

Below is a description of the individual parameters in alphabetical order which are displayed on the Generation Parameters screen:

Parameter	Explanation
CTOUT	Timeout value (in seconds) for changed buffer pool blocks.
DDNAME	Name of the editor work file for the JCL definition.
DSNAME	Name of the work file dataset.
DTOUT	Period of time (in seconds) for logical files to be checked for deletion.
FMODE	Mode of the work file name, which can be from A1 to Z9. This parameter applies under CMS only; it is not displayed in any other environment.
FTOUT	Timeout value (in seconds) for a logical file, which has not been accessed, to be deleted.
IMSG	Initialization message to be issued on the operator console.
ITOUT	Timeout value (in seconds) for the buffer pool initialization.
LRECL	Work file record length.
LTOUT	Timeout value (seconds) for locked buffer pool blocks.
MAXLF	Maximum number of logical files in the editor buffer pool.
PWORK	Percentage of work file records used as work records.
RECNUM	Total number of work file records.
RWORK	Percentage of work records for regular logical files.
UTOUT	Timeout value (in seconds) for unchanged buffer pool blocks.

For further information on these parameters, refer to the section Operating The Software AG Editor (in the Natural Operations for Mainframes documentation).

The Start column refers to the buffer pool restart. The following start values can be displayed for the above parameters:

Value	Explanation
L	The value for the corresponding parameter is taken from either the editor parameter module or the work file definition.
C	A modification of the corresponding parameter value forces a buffer pool cold start. Recovery records are lost.
W	A modification of the corresponding parameter value results in a buffer pool warm start. Recovery records are kept.

Modifying Parameter Values

To modify parameter values

1. Press PF5 (Updat).
2. Press PF5 again to save the new parameter values in the editor work file control record.

They will be activated when the buffer pool is started again.

Users

To invoke the Users function

- On the SYSEDT Main Menu, enter Function Code **U**.
Or, on the SYSEDT Main Menu, press PF7 (Users).

The Users screen is displayed which provides the following information:

Item	Shows
User ID	The Natural user ID.
Logical Files	The number of logical files defined per user.
Pool Blocks	The number of buffer pool blocks per user.
Work Records	The number of work records per user.
Recovery Files	The number of recovery files per user.
Recovery Records	The number of recovery records per user.

The first column contains an input field labeled **M** for Mark. You may enter a question mark (?) in this field to invoke a window which shows you the codes that are valid input for this field.

To perform the function described, you enter the code in the Mark field.

Code	Function
/	Position line to top of screen.
P	
F	Select the logical files of this user.
R	Select the recovery files of this user.
D	Delete all logical files and/or recovery files for this user.

Logical Files

To invoke the Logical Files function

- On the SYSEDT Main Menu, enter Function Code **F**.
Or, on the SYSEDT Main Menu, press PF6 (Files).

The Logical Files screen is displayed which provides the following information:

Item	Shows
File No.	The logical file number.
User ID	The Natural user ID.
Type	The logical file type.
Pool Blks	The number of buffer pool blocks currently used per logical file.
File Recs	The number of work file records currently allocated per logical file.
Last Access	The date and time of the last read or write request per logical file.

In addition, the first column contains an input field labeled **M** for Mark. If you enter a question mark (?) in this field, a window is displayed, which shows you the codes that are valid input for this field.

Enter the code in the Mark field to perform the function described.

Code	Function
/	Position line to top of screen.
P	
S	Select the logical files of this user.
D	Delete the logical file.

Recovery Files

 **To invoke the Recovery Files function**

- On the SYSED T Main Menu, enter Function Code **R**.
Or, on the SYSED T Main Menu, press PF8 (Recov).

The Recovery Files screen is displayed which provides the following information:

Item	Shows
User ID	The Natural user ID.
Member	The library member name.
Library	The library name.
Type	The library type.
Recs	The number of recovery records per recovery file.
Creation Date/Time	The creation date and time of the recovery file.

In addition, the first column contains an input field labeled **M** for Mark. If you enter a question mark (?) in this field, a window is displayed, which shows you the codes that are valid input for this field.

Enter the code in the Mark field to perform the function described.

Code	Function
/	Position line to top of screen.
P	
S	Select the recovery files of this user.
D	Delete the recovery file.

System Administration Facilities

To invoke the Administration Facilities screen

- On the SYSEDT Main Menu, enter Function Code **A**.
Or, on the SYSEDT Main Menu, press PF5 (Admin).

The Administration Facilities screen is displayed which offers you the function of either terminating the editor buffer pool or leaving the SYSEDT utility.

If you choose to terminate the buffer pool (Function Code **T**), a window appears prompting you for confirmation.

If you enter Yes for confirmation, a further window is displayed, asking you whether or not you want the editor buffer pool to be immediately restarted.

If you enter Yes again, the buffer pool is immediately restarted, which gives you the possibility to immediately activate modified generation parameters.

If you specify No, you leave SYSEDT and can perform actions outside your TP environment, for example, change the size of your editor work file; see also Editor Work File and Editor Buffer Pool in the Natural Operations for Mainframes documentation.

SYSRDC Utility

The utility SYSRDC collects data at certain events within Natural. These data can be used for the following purposes:

- Via exit points, the data can be passed to external monitoring and accounting programs for the evaluation of activities in Natural sessions.
- Via a CALL interface, the data can be evaluated in a Natural program to obtain information about the execution flow of a Natural application in the current Natural session. This may be useful for testing purposes.

The data are always supplied at the exit points provided by SYSRDC to be used by external monitoring/accounting programs.

Optionally, the data can also be written into a special SYSRDC buffer. SYSRDC supplies a CALL interface to retrieve the contents of this buffer. With this interface, you can also start and stop the writing of data into the SYSRDC buffer, select events for which data are to be recorded, and write your own events. In this section, the writing of event data into this buffer is referred to as trace recording.

This section covers the following topics:

- Overview
 - Exit Points for External Monitoring/Accounting Programs
 - Trace Recording - CALL Interface
-

Overview

- Data-Collecting Events
- Collected Data
- Activating SYSRDC

Data-Collecting Events

SYSRDC collects data at the following events within Natural (the letters in parentheses are the corresponding event codes used by SYSRDC's CALL interface):

- at session initialization (SI),
- at session termination (ST),
- at program load (PL),
- at program start (PS),
- at program termination (PT),
- before a database call (DB),
- after a database call (DA),
- before a terminal I/O (IB),
- after a terminal I/O (IA),
- before a call of a non-Natural program (CB),
- after a call of a non-Natural program (CA),
- at a runtime error (E),
- at an internal trace call (N), *
- at a user-supplied event (U).

* To activate the "internal trace call" event, the profile parameter ITRACE=ON has to be specified. The Natural components that are to issue internal trace calls have to be defined in the NTTRACE macro of the Natural parameter module (or with the corresponding dynamic profile parameter TRACE). The NTTRACE macro and the parameters TRACE and ITRACE are described in the Natural Parameter Reference documentation.

Collected Data

The collected data can be split in two categories:

- **general data** - which are collected at every event;
- **event-specific data** - which are only collected at specific events.

For the layout of the data, see the macro NAMRDC in the source library, or the local data area RDCDATA in library SYSRDC.

General Data

The following data are collected at every event:

- Software AG product name,
- product version,
- operating system,
- TP monitor,
- run modes (OS/390 systems),
- TP user or batch job name,
- TP terminal ID,
- current Natural user ID,
- current Natural Security user group ID,
- current library,
- current Natural program,
- current program level,
- line number of currently executed statement.

Event-Specific Data

The following data are only collected at the following events:

Event	Data Elements
Session Initialization	none
Session Termination	termination return code termination message ID name of back-end program
Program Load	name of program to be loaded name of load library
Program Start/Termination	program type program name program library name database ID of program library
Database Call	database type command code command ID database ID response code parameter list pointer
Terminal I/O	number of bytes sent number of bytes read total session storage allocated compressed session storage length
Call of Non-Natural Program	name of called program calling mode program link location parameter type parameter address program entry address response code
Runtime Error	error number external abend code error handling program
Internal Trace	up to 250 bytes of information
User-Supplied Event	up to 250 bytes of information

Activating SYSRDC

The SYSRDC utility is activated and controlled by the Natural profile parameter RDCSIZE.

This parameter also determines the size (in KB) of the SYSRDC buffer.

By default, the parameter is set to RDCSIZE=0, which means that the SYSRDC utility is deactivated altogether.

If you set RDCSIZE=2, SYSRDC is activated. The collected data are supplied to the exit points, but are not written into the SYSRDC buffer.

If you wish to activate trace recording (that is, if you wish the data also to be written into the SYSRDC buffer in order to retrieve them within your Natural application via the supplied CALL interface), you have to set the RDCSIZE parameter to a value greater than 2.

To calculate the number of records that fit into SYSRDC buffer, you use the following formula:

$$11 * (RDCSIZE - 2) = \text{number of records}$$

The maximum size of the buffer is 128 KB.

Exit Points for External Monitoring/Accounting Programs

SYSRDC supports any number of concurrent exit points. An exit point can be defined by profile parameter RDCEXIT (see the Natural Parameter Reference documentation). An external monitoring/accounting program can be attached to each exit point.

At every event listed under Data-Collecting Events, the exit points obtain control, using standard linkage call conventions as follows:

Register	Content
1	Points to a parameter address list, consisting of 2 addresses pointing to the general and event-specific data. The layout of these areas is mapped by the DSECT RDCGDATA and RDCLDATA respectively. Both DSECTs are supplied in source form by macro NAMRDC.
13	Points to a 72-byte standard save area.
14	Contains the return address.
15	Contains either the entry point address or the return code of the exit.

Note:

The exits are called independently from the internal CALL interface.

An exit program must have the same attributes as Natural; that is, it must have the same addressing mode, and it has to be reentrant. It must be linked with the Natural nucleus according to the conventions of statically linked non-Natural programs (profile parameters CSTATIC or RCA, as described in the Natural Parameter Reference documentation).

By default, a 400-byte exclusive work area (per session) is supplied for each exit (field RDCGWRKA). If a larger work area is required for an exit, it can be specified after the exit name in profile parameter RDCEXIT. The work area length is passed on to the exits in the field RDCGWRKL and can be used for verification. The location of this work area may change during a session due to Natural relocation, but its contents are preserved.

In teleprocessing environments, the TP anchor address is supplied (field RDCGANCH); for example, the CSA address under CICS. It can be used to access system information.

If a program check occurs during the execution of SYSRDC or of an exit program, further data collecting is disabled for the rest of the session to avoid recursive abend situations.

Return Codes

Non-zero return codes are only supported for two events:

- Before a database call, where Register 15 can contain an Adabas response code which is stored into the control block; the Adabas call will not be executed.
- At program start, where Register 15 can contain a Natural error message number; the program will not be executed but an error condition will occur with the specified number.

Sample Exit Programs

The following sample accounting exit programs are provided in the following source libraries:

Program	Contained in Source Library	Contents
NAMRDC	NATSRCE	DSECT macro for general and event-specific data.
XNATRDC1	NATSRCE	Sample exit program for TSO and OS/390 or VSE/ESA for batch mode.
XNCFRDC1	NATSRCE	Sample exit program for Com-plete.
XNCIRDC1	NCISRCE	Sample exit program for CICS.

Trace Recording - CALL Interface

Trace recording is activated with the profile parameter RDCSIZE and can be used to collect the event data of the current Natural session in a special incore SYSRDC buffer for testing purposes, without using the exits described earlier.

The size of this buffer is determined by the RDCSIZE parameter. The buffer is filled in wrap-around mode; that is, the oldest record is overwritten when the buffer becomes full. At the end of the session, the buffer content is deleted.

Trace recording is started automatically during session initialization and all events are recorded.

The data in the buffer can be accessed by any Natural program within the same session.

Several sample programs, as well as a local data area for layout of the data (RDCDATA), are provided in the library SYSRDC.

Following is information on:

- Program CMRDC
- CMRDC Functions
- CMRDC Return Codes
- Sample Programs in Library SYSRDC

Program CMRDC

The communication between a Natural program and trace recording is performed by invoking the program CMRDC with CALL statements. CMRDC provides the following functions:

- retrieval of trace records,
- start and stop of trace recording,
- selection of event types for trace recording,
- writing of data at user-supplied events.

CMRDC Functions

Retrieval of Trace Records

To read the data from the SYSRDC buffer, you invoke CMRDC with the following statement:

```
CALL 'CMRDC' function event-time gen-data event-data
```

The following parameters are passed:

Parameter	Format/Length	Explanation
<i>function</i>	A1	Possible functions: F Get first trace record. G Get next trace record.
<i>event-time</i>	N10	Time of event (<i>HHMMSSXXXX</i>).
<i>gen-data</i>	A252	General data.
<i>event-data</i>	A252	Event-specific data.

The retrieval functions automatically stop the writing of data into the SYSRDC buffer. To start it again, you use the CALL statement described below.

Start and Stop of Trace Recording

To start or stop trace recording (that is, the writing of data into the SYSRDC buffer,) you invoke CMRDC with the following statement:

```
CALL 'CMRDC' function
```

The following parameter is passed:

Parameter	Format/Length	Explanation
<i>function</i>	A1	Possible functions: S Clear SYSRDC buffer and start trace recording. P Stop trace recording.

Select Events for Trace Recording

By default, all events are selected for trace recording. If you wish only specific events to be recorded, you use this function.

Note:

This function only selects the events at which data are to be written into the SYSRDC buffer. It does not affect the data passed to the exits. It does not affect the status (started/stopped) of trace recording, either.

To select the types of events to be recorded, you invoke CMRDC with the following statement:

```
CALL 'CMRDC' function type...
```

The following parameters are passed:

Parameter	Format/Length	Explanation
<i>function</i>	A1	Possible function: T Select events for trace recording.
<i>type</i>	A1, A2 or A3	Type of event to be recorded: PL Program load PS Program start PT Program termination DB Before database call DA After database call CB Before non-Natural program call CA After non-Natural program call IB Before terminal I/O IA After terminal I/O E Runtime error N Internal trace call NS Statement trace U User trace call ALL All events ' ' (<i>blank</i>) No event

To select multiple events that begin with the same character, you can use asterisk notation. For example, with **P*** you would select all program events (PL, PS and PT).

Write User-Defined Trace Entry

To write a user-defined event into the SYSRDC buffer from within a program, you invoke CMRDC with the following statement:

```
CALL 'CMRDC' function record
```

The following parameters are passed:

Parameter	Format/Length	Explanation
<i>function</i>	A1	Possible function: U Write user-defined trace entry.
<i>record</i>	Annn	Trace record with a length (<i>nnn</i>) of up to 250 bytes.

CMRDC Return Codes

Code	Meaning
0	Function successfully executed.
4	Last trace record (Functions F and G only).
8	Too few parameters for this function.
12	Invalid function code.
16	SYSRDC not active (for example, RDCSIZE=0).
20	SYSRDC disabled after an error.
24	No buffer space available for trace recording (RDCSIZE=2 or smaller; Function U only).
28	Invalid parameter value (Function T only).

Sample Programs in Library SYSRDC

The library SYSRDC contains the following sample programs:

Program	Function
RDCDISP	Sample program to display all records in the SYSRDC buffer showing some selected fields.
RDCSTART	Sample program to start trace recording.
RDCSTOP	Sample program to stop trace recording.
RDCSET	Sample program to select events for trace recording.
RDCUSER	Sample program to write user trace entries.

SYSTP Utility - Overview

SYSTP is used to monitor and control various TP-monitor-specific characteristics of Natural.

To invoke the SYSTP utility

- In the direct command line, enter SYSTP.

A menu is then displayed, from which you can select the following functions:

- General Functions:
 - Natural Monitoring - SYSMON
 - Natural Print/Work Files - SYSFILE
 - Natural Swap Information
 - Buffer Usage Statistics - BUS
 - Natural Subsystems and Roll Server Information
 - Natural Thread Usage Statistics
- Environment-Dependent Functions (not available under Com-plete and TSO)
 - under CICS
 - under IMS/TM
 - under TIAM and UTM

The General SYSTP Functions listed above are available in most environments, under most TP monitors.

The Environment-Dependent Functions available are different for each TP monitor; they are described in the corresponding TP-monitor-specific sections of this documentation.

In batch mode, SYSTP is used to evaluate data which have been collected in Natural online environments:

- SYSTP in Batch Mode

General SYSTP Functions

This section describes the SYSTP functions which are available under most TP monitors:

- Natural Monitoring - SYSMON
 - Natural Print/Work Files - SYSFILE
 - Natural Swap Information
 - Buffer Usage Statistics - BUS
 - Natural Subsystems and Roll Server Information
 - Natural Thread Usage Statistics
-

Natural Monitoring - SYSMON

This function provides statistics related to Natural programs and screen transactions of Natural sessions.

When you invoke this function and the monitoring function is **not active**, a menu is displayed from which you can select the following function:

- Activate Monitor

When you invoke this function and the monitoring function is **active**, a menu is displayed from which you can select the following functions:

- Deactivate Monitor
- Display Monitor Terminal Statistics
- Display Monitor Program Statistics

Activate/Deactivate Monitor

With these functions you can activate or deactivate the monitor function.

When the monitor function is activated, it begins collecting statistical information of current sessions. Once the monitor function is deactivated, a statistical summary is collected and written to the system log file.

Note:

When active, the monitoring function requires additional memory pool space and therefore may affect overall system performance. Set the RDCSIZE parameter to a minimum value of 2 KB (see also Profile Parameters in the Natural Parameter Reference documentation).

Display Monitor Terminal Statistics

Terminal statistics can be displayed for all active terminals or for a single terminal.

- For all active terminals: enter **T** in the Code field.
- For a single terminal: enter **T** in the Code field and the terminal ID in the field Name of LTERM or Program.

The following screen is used to display statistics for all active terminals:

```

17:03:13          ***** NATURAL SYSTP UTILITY *****          2000-11-27
User VR000001    - NATURAL Monitor Terminal Statistics -          TID 0756

Cm Name      Current      NAT- ADA- Ext- Mean-  Screen I/O  User   Sys  Fetch
              time time time  time   No  KB   Acc  Acc
-----
_ 0756      S2SCENT1          0   0   0   0.0    1   0    0    0    5
              0   0   0   .0    0   0    0    0    0
              0   0   0   .0    0   0    0    0    0
              0   0   0   .0    0   0    0    0    0
              0   0   0   .0    0   0    0    0    0
              0   0   0   .0    0   0    0    0    0
              0   0   0   .0    0   0    0    0    0
              0   0   0   .0    0   0    0    0    0
              0   0   0   .0    0   0    0    0    0
              0   0   0   .0    0   0    0    0    0
-----
Function : _          ( + next page / . Exit / ? Help )
-----
Select, mark with function or mark for additional information
Command ==>
Enter-PF1---PF2---PF3---PF4---PF5---PF6---PF7---PF8---PF9---PF10--PF11--PF12---
      Help Menu Exit          +          Canc
    
```

Note:

If the overview of active terminals is displayed repeatedly, an asterisk is set to the terminal most active since the last repetition.

To display statistics for a single terminal, you mark the desired terminal on the screen above.

The following statistics are provided for terminals and programs:

Column	Statistics	Explanation
NAT-time	Time in Natural	Time in Natural nucleus and in the interface.
ADA-time	Time in Adabas	Time waiting for response from Adabas.
Ext-time	Time in external program	Time needed by a user-written module.
Mean-time	Mean evaluation time	Elapsed time of one Natural screen transaction.
Screen I/O No	Number of Screen I/Os	Number of screen I/Os.
Screen I/O KB	Amount of data transmitted	Amount of data transferred to or from the screen.
	Evaluation time > 3 sec	Only applies to statistics for a single terminal. Percentage of evaluation times longer than 3 seconds.
	Evaluation time > 6 sec	Only applies to statistics for a single terminal. Percentage of evaluation times longer than 6 seconds.
User Acc	Number of user file accesses	Counter for accesses to Adabas user files.
Sys Acc	Number of system file accesses	Counter for accesses to Natural system file, including fetches.
Fetch	Number of fetches	Counter for total number of fetches.

Display Monitor Program Statistics

Program statistics can be displayed for all active programs or for a single program:

- For all active programs: enter **P** in the Code field.
- For a single program: enter **P** in the Code field and the program name in the field Name of LTERM or Program, and the library name.

The following screen is used to display statistics for all programs:

```

08:56:53          ***** NATURAL SYSTP UTILITY *****          2000-11-28
User VR000001    - NATURAL Monitor Program Statistics -          TID 0807

Cm Name      Current      NAT-  ADA-  Ext-  Mean-  Screen I/O  User  Sys  Fetch
              time time time  time   No  KB   Acc  Acc
-----
_ SMMEN01 SYSTP          0   0   0  0.0    1   0    0    0    2
_ S2MRAHM1 SYSTP         0   0   0  0.0    0   0    0    0    0
_ S2SCOM01 SYSTP         0   0   0  0.0    0   0    0    0    0
* SMPMEN01 SYSTP         0   0   0  0.0    0   0    3    0    1
_ SMPSTA01 SYSTP         0   0   0  0.0    0   0    0    0    1
_ S2SCENT1 SYSTP         0   0   0  0.0    0   0    0    0    0
              0   0   0  .0     0   0    0    0    0
              0   0   0  .0     0   0    0    0    0
              0   0   0  .0     0   0    0    0    0
              0   0   0  .0     0   0    0    0    0
-----
Function : _          ( + next page / . Exit / ? Help )
-----
Select, mark with function or mark for additional information
Command ==>
Enter-PF1---PF2---PF3---PF4---PF5---PF6---PF7---PF8---PF9---PF10--PF11--PF12---
      Help Menu Exit                               +                               Canc
    
```

For an explanation of the screen output, see the table above.

Note:

If the overview of active programs is displayed repeatedly, an asterisk is set to the program most active since the last repetition.

To display statistics for a single program, you mark the desired program on the screen above.

Natural Print/Work Files - SYSFILE

This function provides information on the available work files and print files.

You can also invoke this function with the system command SYSFILE (Natural Command Reference documentation).

This function can also be used in batch mode.

When you invoke this function, a list of the defined work and print files is displayed, showing the following information for each file:

No.	The number of the work/print file.
Type	The type of assignment; that is, the operating system, TP monitor or Natural product file to which the work/print file is assigned.
Name	The name of the work/print file.
RECFM, LRECL, BLKSZ	The record format, logical record length and block size of the work/print file (if applicable).
Status	The status can be: available for input and/or output, open for input and/or output.

Under VSE/ESA, the logical-unit assignments are also displayed.

With the following keys and commands (which you enter in the first column of the list), you can scroll the list or display additional information:

Key	Command	Function
PF4	S	Displays various additional items of information on the file marked with the cursor/command.
PF5	P	Scrolls the file marked with the cursor/command to the top of the page (same as command /).
	/	
PF6 / PF9		Scrolls to the beginning/end of the list.
PF7 / PF8		Scrolls one page backward/forward.
PF10		Scrolls to the list of print files.
PF11		Scrolls to the list of work files.
	D	Displays the corresponding Natural control block (work file area) in dump format.

Natural Swap Information

This function is only available under CICS and UTM.

The swap pool manager enables online monitoring and control of the Natural swap pool. This section describes how the swap pool manager is used rather than how the swap pool operates. For detailed information on the operation of the Natural swap pool, see *The Natural Swap Pool* in the *Natural Operations for Mainframes* documentation.

From the swap pool manager main menu, you invoke the following functions and sub-functions for the controlling and monitoring the swap pool:

- Administration
- Debugging Facilities
- Information
- Parameter Service
- Status Information

Each of these functions can be invoked by either Function Code or PF key.

Administration

- Slot Size Calculation
- Change Swap Pool Status
- Update Reorg Control Data

Slot Size Calculation

This function displays the optimum values for the layout of the swap pool based on the current usage.

You can store these values to be used for a later initialization/reorganization (once they have been stored, they can also be maintained with the Parameter Service function).

You can also initiate a swap pool reorganization using these values.

For further details, see the online help of this function.

Change Swap Pool Status

This function is used to activate or deactivate the Natural swap pool. In addition, you can modify the wait time and the number of waits for swap pool synchronization.

For further details, see the online help of this function.

Update Reorganization Control Data

With this function, you can modify the most important parameters in swap pool management. To modify the values you must enter a valid password.

For further details, see the online help of this function.

Debugging Facilities

Note:

Do not use this function without prior consultation of Software AG's Natural support.

This function is only available under UTM

With this function it is possible to activate or deactivate an internal screen debugging buffer. Activation of the screen debugging buffer is used to locate terminal I/O inconsistencies if they occur. The function records information on the last three terminal I/O sequences. The buffer has a size of 3 KB and is used in a wrap-around procedure.

In addition, you can activate/deactivate a trace function for asynchronous write operations to the Natural roll file.

For further details, see the online help of this function.

Information

- Show Addresses
- Show Summary of Buffer Usage
- Show Swap Pool Information
- Show Logical Swap Pools
- Show Reorg Control Data
- Show Swap Pool Usage
- Create Statistics List

Show Addresses

This function displays the addresses of various pools.

Show Summary of Buffer Usage

This function is used to optimize the sizes of the various Natural buffers and the Natural user threads (MAXSIZE). It activates, deactivates and displays a summary of Natural buffer usage.

The activation and deactivation of buffer statistics can only be performed with a valid password. For the display of buffer statistics, no password is necessary.

The buffers displayed are the same as those displayed by the function Buffer Usage Statistics.

Show Swap Pool Information

This function displays information on the swap pool currently in use, including control/statistics data, and memory sizes.

The individual items of information shown are explained in the online help of this function.

Show Logical Swap Pools

This function displays the current table of logical swap pools.

On the table, you can mark a specific logical swap pool with any character to get additional information on it.

The individual items of information shown are explained in the online help of this function.

Show Reorganization Control Data

This function displays all information related to the swap pool reorganization.

Displayed in the left half of the screen is the swap pool reorganization table. The table contains cumulative statistics on the comparative sizes between compressed Natural user threads and standard slot size. The table is cleared with each reorganization of the swap pool. The left half of the table shows how often and to what extent the user threads are larger than the standard slot size. The right half of the table shows how often and to what extent the user threads are smaller than the standard slot size. Sizes in this half of the table are expressed in units, which are dependent on the factor specified by the swap pool manager.

In the row labeled **n**, count is taken of user threads which exceed/fall short of the standard slot size by over 9 pages/units. The average length of these user threads is displayed in the row labeled **Av.+n**.

The individual items of information shown are explained in the online help of this function.

Show Swap Pool Usage

This function displays information on the usage of the swap pool since its initialization or the last reorganization.

The individual items of information shown are explained in the online help of this function.

Create Statistics List

This function is used to create a list of the current swap pool usage statistics.

Parameter Service

- Parameter Maintenance
- Password Maintenance

Parameter Maintenance

This function is used to change online the parameters for the initialization or reorganization of the swap pool.

The subfunctions as well as the individual items that can be modified are explained in the online help of this function.

The use of this function is password-protected (see below).

Password Maintenance

This function is used to change or recover the password used for the Parameter Service function.

The initial password is SYSTP.

Status Information

With this function, you can display the current status of the Natural swap pool, of the summary of buffer usage and of the UTM screen debugging.

Buffer Usage Statistics - BUS

This function provides statistical information on the usage of Natural buffers: which buffers are allocated for the current session, and how much buffer space is being used.

The Total figures at the end of the statistics list allow you to draw conclusions about the efficiency of buffer compression.

You can invoke this function either from the SYSTP menu or with the system command BUS.

When you invoke the function, a list is displayed showing all buffers which are actually being used in the current Natural session.

For each of these buffers, the following information is displayed:

M	In this column, you can mark a buffer with a command (see below).
No.	The buffers are numbered sequentially in order of allocation.
Name	The name of the buffer. Only those buffers which have actually been requested in the current Natural session are listed. The buffers are listed under The Individual Buffers.
Type	V indicates a variable buffer. The size of a variable buffer is increased automatically when necessary (even if it is allocated outside the Natural thread). If it is allocated outside the thread, it is copied into the thread at a terminal I/O; if it does not fit into the thread, it is truncated to its actually used length.
Size	The size of the buffer (in bytes).
Used	The number of bytes currently being used. This value is used for buffer compression in environments using threads (for example, CICS or UTM).
Perc. (Used)	The percentage currently being used; that is, the value of the Used column in relation to the value of the Size column.
MaxUsed	The maximum number of bytes which have been used in the course of the current session so far (not the size being used at present).
Perc. (MaxUsed)	The percentage of current session usage; that is, the value of the Max. Used column in relation to the value of the Size column.
MaxSize	The maximum size (in bytes) which been allocated to the buffer in the course of the current session so far (applies to variable buffers only).
Perc.(MaxSize)	The maximum size allocated so far (value of the Max. Size column) in relation to the current size (value of the Size column) (applies to variable buffers only). A percentage of 1000 or more is indicated by 999.9 displayed intensified.
At the end of the list, the following information is displayed:	
ThrdSize	The current size (in KB) of the Natural thread.
Total	The sums of all buffer sizes (in both bytes and KB) and percentages used/allocated. These totals can also be displayed via PF10 (see below). For Max. Size, the total shows the maximum additional amount of thread size that would have been needed in the course of the session so far.

With the following keys and commands (which you enter in the first column of the list), you can scroll the list or display additional information:

Key	Command	Function
PF4	D	Displays the contents of the buffer marked with the cursor/command in dump format (for internal use by Software AG support personnel).
PF5	P	Scrolls the buffer marked with the cursor/command to the top of the page.
	/	
PF6	--	Scrolls to the beginning of the list.
PF7	-	Scrolls one page backward.
PF8	+	Scrolls one page forward.
PF9	++	Scrolls to the end of the list.
PF10		Displays the Total buffer usage figures.
PF11		Displays the relative addresses of the buffers, that is, relative to the input/output control buffer (IOCB).

Individual Buffers

The following table shows the names and functions/contents of all buffers, along with the parameters the buffers are influenced by.

Buffer Name	Function/Contents	Influencing Parameters
ADA_NUKE	Program load buffer	none
ADA_USER	Adabas user call buffer	none
ADASIZE	ADALNK reentrancy buffer	none
AIVDAT	Application-independent variables	none
ASIZE	Entire System Server auxiliary buffer	ASIZE
ASPSIZE	Adabas stored procedures and triggers	none
BB#ESIZE	Main buffer for data and runtime tables	ESIZE
BPMWORK	Buffer pool manager work area	NTBPI macro
BSIZE	Reserved for future use	BSIZE
CFWSIZE	Framework buffer	CFWSIZE
CMPRTSZ	Compression table buffer	none
COMPCGDA	Compiler, GDA Version 2.2 to 2.3 conversion	none
COMPDDMS	DDM loading and generation	none
COMPSYT	Compiler, symbol table	none
COMPTBLS	Compiler, generated object	none
COMPWARN	Compiler, warning messages	none
CONTEXT	Context variables buffer	none
CSIZE	Con-nect buffer	CSIZE
DATSIZE	Local data buffer	DATSIZE
DB2SIZE	Natural for DB2 and Natural for SQL/DS buffer	DB2SIZE
DB2SIZE 1 - 5	Natural for DB2 buffers	none
DLISIZE	Natural for DL/I buffer	DLISIZE
DSIZE	Debug buffer area	DSIZE
EDTPOOL	Editor auxiliary buffer	EDPSIZE
EPLTAB	External program list table	CDYNAM, CSTATIC
ERRMSG	Error messages buffer	none
ESQSIZE	Adabas SQL server buffer	none
ETPSIZE	EntireTransaction Propagator buffer	ETPSIZE
EXCSIZE	Natural Expert C interface buffer	EXCSIZE
EXPAFOBU	Expanded format buffer	none
EXRSIZE	Natural Expert rule tables buffer	EXRSIZE
EXTBUF	Extra Buffer for Adabas Version 4	EXTBUF

Buffer Name	Function/Contents	Influencing Parameters
GETPHTAB	Physical GETMAIN table	none
GLBSYS	System global data area buffer	none
GLBUSER	User global data area buffer	none
HELPBUF	Help message buffer	none
IDIR	Local program directory buffer	none
IDSIZE	IDMS buffer	IDSIZE
IOCB	Driver/frontend buffer	none
IOOATTR	Overlay attribute buffer	none
IOOVLY	Overlay buffer	none
IOPAGE	Page buffer	PS, LS
IOPATTR	Page attribute buffer	PS, LS
IOSATTR	Screen attribute buffer	none
IOSAVE	I/O debugging buffer	none
IOSCRN	Screen buffer	none
ISIZE	Initialization buffer	ISIZE
KAPRIBUF	Kanji buffer	none
MONSIZE	Natural monitor buffer	MONSIZE
NAFSIZE	Natural Advanced Facilities buffer	NAFSIZE
NCPWORK	Command processor buffer	none
NDLADHOC	Natural for DL/I adhoc storage buffer	none
NETBUF	Network buffer	none
NOCBUF	Natural Optimizer Compiler buffer	none
NOCCODE	Natural Optimizer Compiler code buffer	none
NOCSTMT	Natural Optimizer Compiler statement buffer	none
NOCTRACE	Natural Optimizer Compiler trace buffer	none
NOCWBUF	Natural Optimizer Compiler work buffer	none
PRINT 00 - 31	Batch print file buffers	NTPRINT macro
PRNTWORK	Print and work file buffer	NTPRINT macro, NETWORK macro
PROFBUF	Natural profile work buffer	none
PSEUDORV	Pseudo register vector buffer	none
RCEXEC	Object runtime command buffer	none
RDCSIZE	SYSRDC buffer	RDCSIZE
REVWSIZE	Review buffer	none
RJESIZE	NATRJE buffer	RJESIZE
RPCSIZE	Remote procedure call buffer	none

Buffer Name	Function/Contents	Influencing Parameters
RSPOOL	Remote spool buffer	none
RUNSIZE	Runtime buffer	RUNSIZE
SFEXEC	RUN command execution buffer	none
SFSRCALF	System file interface buffer	none
SFSRCMF	System file interface buffer	none
SFSRCUS	System file interface buffer	none
SFSYT	System file interface buffer	none
SORTESM	External sort manager buffer	none
SORTSIZE	Sort buffer	NTSORT macro
SPEXEC	Sysplex execution buffer	none
SSIZE	Buffer for NSPF editor	SSIZE
SUREBUF	Setup/return information buffer	none
SWPSSIZE	Swap pool statistics work buffer	none
TESTWORK	Test utilities buffer	none
TIOB	Terminal I/O buffer	none
TISIZE	SYSPARM parameter buffer	none
TRMIODEB	Terminal I/O debugging buffer	none
TSIZE	Text retrieval buffer	TSIZE
UDSSIZE 0 - 5	UDS buffers	none
USERBUF	User buffer	USERBUF
VSIZE 0 - 9	Natural for VSAM buffers	VSIZE
WORK 01 - 32	Batch work file buffers	NTWORK macro
WSISIZE	Natural Workstation Interface buffer	WSISIZE
WSIZE	Screen images buffer	WSIZE
XSIZE	Natural Connection buffer	XSIZE
ZSIZE	Entire DB buffer	ZSIZE

Natural Subsystems and Roll Server Information

This function is only available under OS/390.

You can use it to determine an optimum thread size or roll file size for a Natural application. It displays a list of the Natural subsystems together with the current status of the related authorized service manager and roll server.

The following commands are available for each listed subsystem:

Command	Function
B	Displays buffer pool information (name, size, type).
R	Displays roll server statistics.
S	Displays zaps applied to the authorized service manager.
Z	Displays zaps applied to the roll server.
L	Displays and resets entries in the roll file directory.

This information is useful for tuning the roll server, as described under Roll Server in the Natural Operations for Mainframes documentation.

Natural Thread Usage Statistics

This function is only available under CICS, Com-plete, IMS/TM and UTM. It is not available in a Sysplex environment.

This function allows you to determine an optimum thread size or roll file size for a Natural application.

You should activate this function only when needed, and deactivate it after you have determined your optimum thread size, because this function occupies space in the Natural buffer pool. When you deactivate it, the space in the buffer pool becomes available again.

Proceed as follows:

1. Define an oversized thread in the range of 512 to 1024 KB for your Natural application. Take into account the number of Software AG subproducts used.
2. Start your Natural application, either in production or in test mode.
3. Activate the Natural Thread Usage Statistics function: Invoke the SYSTP utility. On the SYSTP main menu, choose function T (Natural Thread Usage Statistics). On the menu that appears then, choose function A (Activate Statistics).
4. Use your Natural application under typical production conditions. The Thread Usage Statistics function runs in the background and logs the buffer sizes used.
5. Then invoke the SYSTP Thread Usage Statistics function again. On the menu that appears, choose function S (Show Statistics), P (Print Statistics) or D (Deactivate and Print Statistics). It is recommended that you use function D to free buffer pool space.

The Natural Thread Usage Statistics contain the following information:

Ext. Buffer	The sizes of these buffers are defined externally (in the Natural parameter module).
Defined Size	The buffer size as defined in the Natural parameter module.
Max. Allocated Size	The maximum buffer size allocated. Note that for the internal BB area, 14368 bytes are added to the ESIZE profile parameter value.
Max. Used Size	The maximum buffer size used.
Sum of External Buffer Sizes	The total of all buffer sizes defined in the Natural parameter module.
Sum of Internal Buffer Sizes	The total of all buffer sizes requested by Natural internally.
Max. Used Thread Length	The maximum thread length used by Natural. Define this length as your minimum (optimum) Natural thread length. Round it up to the next KB number that can be divided by 2.
Max. Compressed Thread Length	The maximum length of a compressed Natural thread that was written to the Natural roll file. Define this length as your minimum (optimum) Natural roll file length.

Show Physical GETMAIN Statistics

The physical GETMAIN statistics provide information on all physical GETMAINs relevant for the Natural work pools and the variable Natural buffers outside the Natural user threads.

They indicate the original buffer sizes (during the startup of a Natural session), the number of physical GETMAINs, the buffer length for the physical GETMAIN and the buffer position (above or below the 16-MB line).

The statistics data always refers to the buffers with the greatest lengths requested within a terminal I/O, for all users of the Natural application.

The statistics provides a maximum of six entries for each buffer. These entries may be overwritten through the wrap around procedure. The highest number equals the maximum number of the physical GETMAINs within a terminal I/O, for each buffer concerned.

The first two entries in the statistics refer to the Natural work pools (if available) above (WRKPOOLA), respectively, below (WRKPOOLB) the 16-MB line.

Here, the highest physical GETMAIN number refers to the amount of work pools simultaneously available during the terminal I/O. The sum of all work pool lengths amounts to the total storage requirement of the work pools within a terminal I/O.

All subsequent statistics entries refer to the physical GETMAINs for the variable Natural buffers, which either could not be defined in the Natural user thread due to insufficient space, or were increased outside the Natural user threads. For these buffers, the highest physical GETMAIN number indicates the greatest space requirement for each buffer within a terminal I/O. The total storage space requested earlier was deallocated before each of the following physical GETMAINs.

That is, the sum of all physical GETMAINs with the highest number shows the maximum storage requirement for the variable buffers outside the Natural user threads during a terminal I/O, for all users of the Natural application.

SYSTP Functions under CICS

Under CICS, SYSTP provides the following environment-dependent functions:

- Natural User Sessions
- Natural Roll Facilities
- Natural Thread Groups
- Natural Storage Threads
- NCI Global System Information
- NCI Generation Options
- Natural Thread Group Definitions
- Own Natural User Session
- CICS Task Information
- System Administration Facilities
- Applied NCI ZAPs

For many of the screens accessible from this menu, the first column contains an input field labeled **M** for Mark. For all such screens, the following codes are valid input. Enter the code in the Mark field to perform the function described.

Code	Function
.	Exit screen and return to previous level
/	Position line to top of screen
P	
S	Display detailed information on selected line

Help information can be invoked for each function by pressing PF1.

If you enter a question mark (?) in the command line, all direct commands available within the SYSTP utility under CICS are displayed.

Natural User Sessions

This function displays an overview of user sessions in the Natural environment.

For each session, the following information is displayed:

TID	Terminal ID unique within CICS.
User ID	User ID used to sign onto CICS.
Tran	CICS transaction ID under which Natural session is currently running. For pseudo-conversational sessions, this is the pseudo-conversational restart transaction code.
Start Date/Time	Starting date and time of the Natural session.
Last Act	Time of last screen output.
Status	User work area status.
Program	The Natural program currently active.
Logon ID	The Natural library in which the user is currently working.

If you press PF10, the display of the session date and time is replaced by the following session resource data:

Thrd Grp	Thread group to which user is assigned.
Thread	Name of thread last used.
Roll Fac	Assigned roll facility.

The first column of the screen is an input field labeled **M** for Mark. In this field, you can enter any one of the following codes:

Code	Function
C	Cancel session. At next input, session is terminated and user is informed.
F	Flush session. Session is terminated at once. Session is marked and labeled with operational status Purged by Administrator and resources are made available to other users immediately.
R	Reactivate session. This function can be used to reverse the cancelling of a session if user has not entered input.

User Session Statistics

For each session displayed in the user session screens shown above, additional information can be displayed by invoking the screen Natural User Session Statistics. This screen is invoked by either placing the cursor in the Mark field for a session and pressing PF4 or entering code **S** or **U** in the Mark field for a session and pressing ENTER.

The Natural User Session Statistics screen is displayed, showing the following information for each user session:

Note:

All sizes on the Natural User Session Statistics screen are in KB unless otherwise indicated in the field descriptions below.

Session Start	Day, date and time when the session was started.
Last Actions	Date and time the user was active last.
User	ID of the user (*USER).
Terminal	ID of the terminal associated with the Natural session (*INIT-ID).
Transid	(Pseudo-conversational) transaction ID under which Natural is running.
Cur Strg Used	Current amount of storage used by this session.
Max Strg Used	Maximum amount of storage ever used by this session.
Common Thread Size	This group's common thread size.
Thread Group	Name of the associated Thread group (triggered by starting the transaction ID).
Max Strg Used	Maximum amount of storage ever used by any user in group.
Thread Size	Size of this thread.
Thread Name	Name of the thread used last. For threads allocated via GETMAIN, the thread name is composed of the prefix NSCP followed by the terminal ID.
Max Strg Used	Maximum amount of storage ever used in this thread.
Natural Logon ID	Natural application ID (*LIBRARY-ID).
Natural Program	Name of the Natural program currently used by the session (*PROGRAM).
Operational Status	See the table below.
Session Resumes	Total number of session resumes.
Swap-Ins	Number of session resumes with swapping in from swap pool.
Thread Switches	Number of session resumes with swapping/rolling into a thread which is different to the one the session had been in before.
Roll-Ins	Number of session resumes with rolling in from roll facility.
Roll Facility	Name of associated roll facility.
Roll Recs (Last)	Number of records written to roll facility for last roll-out.
Roll Recs (Max)	Maximum number of records ever written during roll-out.
Roll Record Size	Record size of this roll facility.
Slot Size	Number of records required to roll-out a thread completely.
Restart Rec. No.	Number of the record that contains roll-out control information; this record must be rolled in first.
Compressed Length	Amount of relevant storage currently swapped/rolled out.
Slot Number	Number of slot in VSAM roll file belonging to this session (for VSAM only).

VSAM Roll Files

The following information applies to VSAM roll files only:

The relationship between restart record number (RecNum), slot number (SN) and slot size (SZ) is:

$$\text{RecNum} = (\text{SN}-1) * \text{SZ} + 2 \quad \text{or} \quad \text{SN} = (\text{RecNum}-2) / \text{SZ} + 1$$

A session can assume the following operational states:

Status	Abbreviation	Description
Active	Act	Currently active
Inactive	Ina	Inactive, still in thread
Swapped	Swp	Swapped, in swap pool
Rolled out	Rld	Rolled out, in roll facility
Wait (Init)	WtI	Waiting for thread on session initialization
Wait (Resume)	WtR	Waiting for thread on session resume
Initializing	Int	Initializing session
Resuming	Res	Resuming session, in thread, not active yet
Suspending	Sus	Suspending session
Terminating	Trm	Terminating session
Swapping out	Swo	Session swapping out
Swapping in	Swi	Session swapping in
Rolling out	Out	Rolling out from thread or swap pool
Rolling in	In	Rolling in from roll facility

The following additional information can appear in the status line:

Status	Description
Conversational	Dialog-oriented session (PSEUDO=OFF) as opposed to pseudo-conversational/transaction-oriented session.
Forced Conversational	Last screen I/O of a PSEUDO=ON session was conversational.
No-Roll	Session is not allowed to roll.
Compressed	Session is compressed (in swap pool or roll facility).
Thread Switched	The thread currently used is not the same as used before.
Thread Locked	Session kept from switching threads (for example, RELO=OFF); may also force No-Roll/Conversational status.
Purged by Administrator	Session cancelled by administrator (flag set).
Spool Task	The task is a spool/print task.
Asynchronous Task	The task is an asynchronous task, not bound to a terminal.

Natural Roll Facilities

This function is used to display which swap files are available for rolling out user work areas to make room in the swap pool for active users. These swap files are known as *roll facilities*.

When you invoke this function, the Natural Roll Facilities screen appears, displaying the following information for each roll facility:

Facility Name	TEMPSTOR is used for auxiliary temporary storage, MAINSTOR for main temporary storage, and remaining file names are VSAM roll files as defined in the CICS file control table (FCT).
Record Size	Record Size of this roll facility.
Slot Size	Number of records required to roll out a thread completely (maximum thread size divided by record size, rounded up).
No. of Slots	Number of sessions which fit into this roll file (number of file records divided by slot size, rounded down); applies to VSAM roll files only.
Facility Users	Current/maximum number of user sessions assigned to this roll facility.
Roll Counts	Number of session roll operations into or from this roll facility.
Status	Indicates Full if the facility users equal the number of available slots.

Natural Thread Groups

This function is used to display which thread groups are available to Natural.

For each thread group, the following information is displayed:

Group Name	Thread group name.
Group Users	Current and maximum number of users assigned to this thread group.
Thread Type	<p>SHR Permanent threads allocated via GETMAIN SVC or CICS GETMAIN SHARED are used.</p> <p>GETM Threads allocated via GETMAIN are used.</p> <p>none No threads are used by transactions defined in this thread group, GETMAINs are passed to CICS.</p>
TCBs	Maximum number of sessions concurrently active.
Thread Size	Thread group's common thread size.
Strg Used	Maximum storage allocated to any thread in this group.
Queue Sizes	<p>The current and maximum queue size for the thread group's central wait queue and the number of times the maximum was reached.</p> <p>Only applies if the parameter THREADS has been defined as greater than zero for this thread group.</p>
VSAM/Aux/Main	Roll facilities defined for group; TEMPSTOR always backs up VSAM if VSAM roll files are not available or full.

To display additional information on a thread group, enter one of the following codes in the Mark field or place the cursor in the Mark field and press the appropriate PF key:

Code	Key	Function
T	PF10	Display program storage threads in thread group.
D	PF11	Display thread group definition.

Natural Storage Threads

This function is used to display information on the storage threads in the Natural environment.

For each thread, the following information is displayed:

Thread Name	Name of the thread.
Grp No.	Number of the group to which this thread belongs.
Thrd Size	Usable Thread size.
Strg Used	Maximum amount of storage ever used in this thread.
Use Count	Number of times this thread has been selected for processing.
Roll-Ins	Logical: Session resumes. Physical: Roll-in from roll facility.
Queue Sizes	Current number of users queuing on thread. If this number n is greater than 1, n minus 1 users are waiting; maximum queue count for this thread; number of times at maximum.
Term ID	Terminal ID belonging to the Natural session whose data are in thread.
Task No.	ID of CICS task currently active in this thread. If no ID is displayed, no session is active in this thread.

To display additional information on a thread group, enter one of the following codes in the Mark field or place the cursor in the Mark field and press the appropriate PF key:

Code	Key	Function
G	PF10	Display thread group.
D	PF11	Display thread group definition.

NCI Global System Information

This function is used to display data on the system directory.

When you invoke this function, a screen with the following information is displayed:

Natural User Sessions	Current and maximum number of Natural sessions in the system.
Concurrent SCP Active	Current and maximum number of concurrent system control program (SCP) requests (INIT/SUSP/RESM/TERM).
SIR Block Extensions	Current and maximum number of local SIR block extensions.
Slots in 1st SIR Block	Number of user sessions which fit into the primary user control block (first USERS subparameter in NCMDIR macro).
Slots in SIR Blk. Extns.	Number of user sessions which fit into a secondary user control block (second USERS subparameter in NCMDIR macro).
VSAM Roll File Slots	Number of VSAM roll files to check (ROLLFLS).
Possible Roll Facilities	Number of VSAM roll files plus two for CICS TEMPSTOR.
Thread Groups	Number of thread groups determined by evaluating all NCMTGD macro specifications at system startup.
System Recoveries	Number of corrections of statistics counts and/or control block chain.
Size of DIR Extension	Number of bytes used at system startup for thread control blocks and VSAM roll file online directories.
Available Resources	Type, size (in KB), definition - as a program or via GETMAIN (GETM) - and location (below or above the 16 MB line) of all supported buffer pools.
Max Thread Size	Largest thread size across all valid thread groups.
VSAM Roll Files	Indicates whether VSAM roll files are available; that is, existing in VSAM, formatted and defined in the CICS FCT.
Aux TempStor	Indicates whether CICS auxiliary temporary storage is available for the Natural/CICS roll facilities.
Main TempStor	Indicates whether CICS main temporary storage is available for the Natural/CICS roll facilities.
Session Destination	Indicates whether the Natural/CICS log destination (as defined with the LOGDEST parameter of the NCMPRM macro) is defined in the CICS DCT (destination control table) and available.
Message Logging	Indicates whether the Natural/CICS error message destination (as defined with the MSGDEST parameter of the NCMPRM macro) is defined in the CICS DCT and available.
Message Switching	Indicates whether the message switching transaction ID (as defined with the MSGTRAN parameter of the NCMPRM macro) is available and a valid transaction code. Note: If this transaction code is not available, a SYSTP session flush is not possible.
Trace Active	Indicates whether the Natural/CICS trace function is currently active; see also System Administration Facilities.

NCI Generation Options

This function is used to display generation parameter settings for Natural running under CICS. The values of these parameters are determined in the macro NCMPRM, which is part of the Natural/CICS parameter module created during installation.

When you invoke this function, an overview of the generation option settings for Natural will be displayed.

Behind each parameter setting in the Generation Options screen is a parameter of the NCMPRM macro. These parameter names can be viewed by pressing PF10. Use PF10 to toggle between the screen containing the parameter names and explanations of the parameters.

Natural Thread Group Definitions

This function is used to display Natural thread group definitions.

For each thread group definition, the following information is displayed:

Grp No.	Thread group number.
Group Type	Type of group definition: SHR Permanent storage threads to be used/loaded for thread group. GETM Storage threads to be GETMAINED (AMODE31-eligible operating systems only). NONE No threads to be used; all Natural storage requests are passed to CICS. ALIAS Thread group redefinition to assign other primary roll facility triggered by transaction code/task request key.
Roll Fac.	Primary roll facility assigned: VSAM, TEMPSTOR, MAINSTOR or "none".
Thread Size	Thread storage GETMAIN size (for thread group types GETM and SHR).
No. Thrds.	Maximum number of Natural sessions concurrently active in this thread group.
Transaction Codes	As defined in CICS PCT.
Task Request Keys	As defined in CICS PCT.

Own Natural User Session

This function displays data about your current Natural session.

The output is the same as the one for the function User Session Statistics.

CICS Task Information

This functions displays status information on the Natural task in a CICS environment.

System Administration Facilities

This function is used to access facilities for debugging and tracing.

When you invoke this function, a menu is displayed which offers you the following functions:

- Trace Facilities (reserved for future use),
- Debugging Facilities (reserved for future use),
- System Snapshot for Logging,
- Reset System Highwater Marks.

System Snapshot for Logging

This function provides complete SYSTP batch reports (see also SYSTP in Batch Mode) with information on all SCP facilities, regardless of whether they have been used or not. Such facilities are:

- thread groups,
- TYPE=SHR threads,
- roll facilities.

All this information is logged to the Natural/CICS log file (if available).

Reset System Highwater Marks

This function comprises the system snapshot function previously described. In addition, all system highwater marks can be reset, for example:

- the number of user sessions,
- every thread group and roll facility,
- the number of UCB block extensions,
- the amount of storage,
- all thread groups and TYPE=SHR threads,
- all wait queue values and counts,
- all roll facility roll counts.

Applied NCI ZAPs

This function displays the numbers of all ZAPs that have been applied to the current Natural TP environment.

SYSTP Functions under IMS/TM

Under IMS/TM, SYSTP provides the following environment-dependent functions:

- Broadcasting
 - Display Environment Data
 - Monitoring
 - Multi Session
 - Applied NII ZAPs
-

Broadcasting

This function is used to broadcast messages to specific user groups in an IMS environment.

When you invoke this function, a menu is displayed from which you can select the following functions:

- Create Broadcast Message
- List all Broadcast Messages

For details on the broadcasting function, see the section Natural under IMS/TM - Special Functions in the Natural TP Monitor Interfaces documentation.

Display Environment Data

This function is used to display environmental data on Natural IMS/TM.

When you invoke this function, the current parameter settings of the Natural IMS/TM interface are displayed.

The parameters cannot be updated. For more information on IMS parameters, see the section Natural under IMS/TM in the Natural TP Monitor Interfaces documentation.

Monitoring

This function is used to monitor Natural sessions in a single swap pool.

When you invoke this function, a menu is displayed from which you can select the following functions:

- Display active Sessions
- Display suspended Sessions
- Display specific monitoring Data (search for sessions based on search criteria).

Multi Session

This function is used to manage multiple Natural sessions.

When you invoke this function, the active sessions are listed. Additionally, the setting of the CREATE and RESUME keys is displayed.

Applied NII ZAPs

This function displays the numbers of all ZAPs that have been applied to the current Natural TP environment.

SYSTP Functions under TIAM and UTM

Under TIAM and UTM, SYSTP provides the following environment-dependent functions:

- P-Key Utility
- Show Common Memory Pools

P-Key Utility

This function supports the loading of programmable P keys on Siemens 975X terminals under the TP monitors TIAM and UTM.

This function allows you to either load the standard Natural key settings (function-key mode KN, KO or KS) to the keys P1 to P20 or to load user-defined values to individual keys.

When you invoke the P-Key utility, the following menu appears:

```

15:54:05          ***** NATURAL SYSTP UTILITY *****          1998-03-25
User VR000001          - P-Key Utility -          TID 0709

                Code   Function                Parameter

                KU    Load User Values    A,H
                KS    Set KS Mode         L,N
                KN    Set KN Mode         L,N
                KO    Set KO Mode         L,N
                KF    Load F1 - F20
                ?     Help
                .     Exit

                Code .. __                Parameter A

Select function.
Command ==>
Enter-PF1---PF2---PF3---PF4---PF5---PF6---PF7---PF8---PF9---PF10--PF11--PF12---
                Help Menu Exit KU    KS    KSN  KN    KNN  KO    KON  KF    Canc
    
```

On this menu, you enter a function code and an optional parameter code. The valid parameter codes for a function are listed to the right of the function. The various codes have the following meaning.

Parameter	Meaning
A	Values are entered in alphanumeric format.
H	Values can be entered in alphanumeric or hexadecimal format.
L	Load option. Mode is set and P keys are loaded.
N	No-load option. Mode is set, but P keys are not loaded.

Set Key Assignment Mode

The following functions are used to set key assignment modes on Siemens terminals:

Set KS Mode	This function executes the terminal command %KS and is invoked either by pressing PF5 or by entering Code S on the menu of the P-key utility.
Set KN Mode	This function executes the terminal command %KN and is invoked either by pressing PF6 or by entering Code N on the menu of the P-key utility.
Set KO Mode	This function executes the terminal command %KO and is invoked either by pressing PF7 or by entering Code O on the menu of the P-key utility.

For a full explanation of key assignment modes, see Natural under BS2000/OSD in the Natural Operations for Mainframes documentation.

Loading Send-Key Codes to P Keys

This function is used to load specific send-key (F) codes F1 to F20 to the keys P1 to P20. The function is similar to the key assignment mode KN, except that F codes can be selected individually.

When this function is invoked, the following screen appears:

15:56:34	***** NATURAL SYSTP UTILITY *****				1998-06-25
User VR000001	- Load F-Codes -				TID VR000001
P01 _	P02 _	P03 _	P04 _	P05 _	
P06 _	P07 _	P08 _	P09 _	P10 _	
P11 _	P12 _	P13 _	P14 _	P15 _	
P16 _	P17 _	P18 _	P19 _	P20 _	
Mark P-Key to be loaded with F-Code					
Command ==>					
Enter-PF1---PF2---PF3---PF4---PF5---PF6---PF7---PF8---PF9---PF10--PF11--PF12---					
Load	Menu	Exit			Canc

To load P keys with F codes, mark the appropriate keys and press ENTER. Only the keys which are marked are loaded with F codes. Other P keys retain their original values.

User Exit LPFSUP01

The Load User Values function is also available to user applications as a user exit. The user exit consists of the Natural subprogram LPFSUP01, which performs the loading of the keys. LPFSUP01 can be copied into user libraries or steplibs.

LPFSUP01 is called as follows:

```
CALLNAT 'LPFSUP01' P-VALUE(*)
```

P-VALUE must be defined as an array: (A24/20).

Example:

```
DEFINE DATA LOCAL
  1 P-VALUE (A24/20)
END-DEFINE
* LOAD '/STA L EM DUE1' TO P1, '/STA P EM DUE1' TO P4
COMPRESS '/STA L' h'192786' INTO P-VALUE(1)
COMPRESS '/STA P' h'192786' INTO P-VALUE(4)
CALLNAT 'LPFSUP01' P-VALUE(*)
END
```

See also the example LPFEXAM1 in library SYSTP.

Show Common Memory Pools

This function displays a list of all common memory pools used in Natural.

The individual items of information shown for each common memory pools are explained in the online help of this function.

SYSTP in Batch Mode

SYSTP can also be used to obtain statistical data on Natural/CICS sessions in batch mode.

The Natural log file to which the statistical data are written must be assigned to the Natural batch job as work file 1 (that is, via CMWKF01). It must also be defined to the online system, which means in the CICS DCT (destination control table); see the LOGDEST parameter in macro NCIPARM of the Natural/CICS parameter module in section Natural under CICS in the Natural TP Monitor Interfaces documentation.

- Invoking SYSTP in Batch Mode
 - Evaluating the Log File
-

Invoking SYSTP in Batch Mode

To invoke the SYSTP utility in batch mode, you specify either of the following commands in the batch job:

- SYSTP *xxx*
- LOGON SYSTP
SYSBATCH *xxx*

where *xxx* indicates what kind of data are to be processed; *xxx=nci*, for example, would indicate that the data were collected by a Natural/CICS online system.

Evaluating the Log File

Data are written to the Natural log file whenever Natural is started, a Natural session is terminated, or the Natural swap pool is reorganized.

The SCP system writes the following records to the Natural log file:

- a start log record whenever the SCP system is initialized,
- a session log record whenever a Natural session is terminated,
- a reorganization log record whenever the swap pool has been reorganized.

When an SCP environment is initialized, a System ID is written into the system control block. This system ID also belongs to all log records. Therefore, a Natural log file can be shared by several SCP online environments.

The information logged serves to keep track of the usage of the SCP online environment. Therefore, most of the information refers to facilities of the SCP environment. The log file is not intended to be an accounting or monitoring tool that refers to facilities of CICS.

Based on the system ID, several reports are created with data related to a Natural session:

- log file data listed in chronological order, which means that session log records are sorted by session end date and time;
- statistical information on how the SCP environment was set up and used;
- statistics on thread groups (if used);
- statistics on program storage threads (if used);
- statistics on roll facilities (if used);
- reorganization statistics (in the case of swap pool reorganizations)

This set of reports is created for all SCP systems with records on the Natural/CICS log file.

Note:

The session termination log records, of course, reflect only resources which have been used by the corresponding sessions. Therefore, these records may not reflect the full SCP environment. Reports of a full SCP environment can be obtained by making a snapshot of the whole SCP environment using the System Administration Facilities.

Sample Batch Job - OS/390:

```
//NATLOG JOB (user,,999),CLASS=K,MSGCLASS=Z
//NATBATCH EXEC PGM=natbatch,REGION=2000K,
// PARM=( 'DBID=nn,FNR=nn,IM=D,MT=0,MADIO=0,AUTO=OFF,MENU=OFF' )
//STEPLIB DD DISP=SHR,DSN=natural.loadlib
//DDCARD DD *
ADARUN DA=nn,SVC=nnn,DE=3380
//CMPRINT DD SYSOUT=A
//CMWKF01 DD DISP=SHR,DSN=nat.log.file
//SYSOUT DD SYSOUT=X
//SYSUDUMP DD SYSOUT=X
//CMSYNIN DD *
SYSTP NCI
/*
```

Sample Batch Job - VSE/ESA:

```
* $$ JOB JNM=NATLOG,CLASS=0,DISP=D,PRI=3
* $$ LST CLASS=A,DISP=D,COPY=1,RBS=100,DEST=*
// JOB NATLOG
// DLBL CMWKF01, 'nat.log.file'
// EXTENT SYS001,xxxxxx
// ASSGN SYS001,DISK,VOL=xxxxxx,SHR
// ASSGN SYSLST,uuu
// LIBDEF PHASE,SEARCH=...
// EXEC natbatch,SIZE=natbatch,PARM='SYSRDR'
DBID=nn,FNR=nn,IM=D,MT=0,OBJIN=R,MADIO=0
AUTO=OFF,MENU=OFF,BWORKD=(1,1,4628,VB)
/*
ADARUN DA=nn,SVC=nnn,DEVICE=3380,TNAE=999999,TT=999999,MODE=MULTI
/*
SYSTP NCI
/*
// EXEC LISTLOG
/&
* $$ EOJ
```

NATPAGE Utility - Screen Paging

The screen paging utility NATPAGE can be used to record output data (maps and reports) during a Natural session. The term screen in this context means the contents of the page buffer; that is, the logical page output by Natural.

The screens recorded are stored in the Natural scratch-pad file as described in the relevant section in the Natural Operations for Mainframes documentation.

The maximum number of screens that can be recorded is determined by the session parameter PD, which is described in the Natural Parameter Reference documentation.

Note:

While the NATPAGE utility is used to solely record output data, the Natural Recording utility is used to record input data and keyboard actions as described in the relevant documentation.

The NATPAGE utility consists of the following Natural terminal commands:

Command	Function
%P	Activates NATPAGE and records the contents of the current screen and all subsequent screens. Screens recorded previously are deleted.
%I	Activates NATPAGE and records the contents of the current screen.
%O	Deactivates NATPAGE.
%S	Resumes NATPAGE.
%E	Displays the screens recorded with NATPAGE.

See the Natural Programming Reference documentation for a detailed description of these terminal commands.

Natural Recording Utility

With the Natural Recording utility, you can record a Natural session and later play back the recorded session.

Note:

While the Natural Recording utility is used to record input data and keyboard actions, the NATPAGE utility (see the relevant section) is used to solely record output data as described in the relevant documentation.

Related Documentation: Terminal Commands, Natural Programming Reference.

This section covers the following topics:

- Purpose of Recording
 - Data and Functions Recorded
 - Recording a Session
 - Playing Back a Recording
 - Manipulating a Recording
-

Purpose of Recording

The Natural Recording utility can be used for the following purposes:

- **Demonstration**
Instead of having to type in several commands, such as input data by hand, you can play back a recorded sequence of keyboard actions to demonstrate a standard procedure.
- **Application development**
When applying the same modifications to several objects (for example, programs or maps), you can use a recording to reduce the amount of work involved and at the same time ensure that the modifications are actually the same for all objects affected.
- **Testing**
You can execute a standard testing procedure by simply playing back a recording.
- **Quality control**
Before and after making changes to an application, you can play back a recording and compare the results of the two runs to make sure that certain things were not affected by the changes.
- **User training**
You can incorporate the playback of recordings into training programs for users, to show them specific procedures. Also, you can record users' keyboard actions in a session and then inform them of any errors they make or of ways to carry out actions more efficiently. The recording of users' actions can also help you to detect any flaws in an application's user interface.

Data and Functions Recorded

The Natural Recording utility records:

- All input data and commands (including terminal commands) entered on the screen,
- Any function keys pressed.
- The current cursor position as contained in the Natural system variable *CURSOR.

Recording a Session

Below are the steps required to activate and deactivate a recording:

- Specifying Libraries
- Activating a Recording
- Deactivating a Recording

Specifying Libraries

▶ To specify the library in which all subsequent recordings are to be stored

- Enter the terminal command `%B=library-name`.

If you activate the recording process without having specified *library-name*, the name of the library in which the recording is stored is the same as the value of the system variable `*INIT-USER` at the time when the recording process is activated.

When you log on to another library during a session being recorded, the library in which the recording is being stored remains the same (that is, either the one specified with `%B=` or the `*INIT-USER` library); this means that one recording can record keyboard actions across multiple applications.

Activating a Recording

▶ To activate a recording

- Enter the terminal command `%Bname`.

All subsequent keyboard actions are recorded.

Name denotes the name under which the data recorded are saved in source form as a Natural object of the type Recording. You can treat this source as any other Natural source (for example, delete it, copy it), except that you must not edit it: recordings contain binary data an editor will destroy.

Name can only be specified once. If a recording object of the same name already exists in the library specified for recording, Natural returns the message "Error in recording activation".

Attention:

Any situation that leads to a backout transaction or rollback (for example, a non-activity timeout) while a recording is in progress, will delete part of the recording thus making the entire recording useless.

Terminal Command `%Aname` included in a recording should be followed by Terminal Command `%B` as described in Recording %A below.

Deactivating a Recording

▶ To deactivate a recording

- Enter the terminal command `%B`.

The recording has terminated.

Playing Back a Recording

When a recording is played back, the sequence of, for example, commands and function keys is actually executed again.

The recording is independent of the terminal type; that is, a session recorded on one terminal can be played back on a terminal of another type. You can also play back a recording in batch mode; a recorded online session may, of course, react differently when played back in batch.

Below is information on:

- Step Mode and Background Mode
- Activating a Playback
- Interrupting a Playback

Step Mode and Background Mode

A recording can be played back in two modes: background mode and step mode.

In background mode, the entire recording is played back invisibly; that is, all keyboard actions of the recording are carried out without anything being displayed to you on the terminal screen during the execution of the recording. You cannot interrupt a recording that is played back in background mode, unless the recording contains the terminal command `%R` as explained below.

In step mode, a recording is played back step by step and all keyboard actions are displayed on the screen. By choosing any function key, you proceed from one step to the next. In step mode, it is also possible for you to interrupt the recording by pressing `CLEAR` as explained below.

By default, a recording is played back in background mode.

To set modes

- Enter the terminal command `%GON` to activate step mode.
- Enter the terminal command `%GOFF` to deactivate step mode and activate background mode.
- Enter the terminal command `%G` to toggle between step and background mode.

Activating a Playback

To play back a recording

- Enter the Natural terminal command `%Aname`.

The recording saved under the specified name is executed again.

Recording `%Aname`

If you issue the command `%Aname` while a session is being recorded, the recording specified with `%Aname` is not executed but the command `%Aname` is included into the object source that is being recorded. Thus, you can execute a recording from within another recording and concatenate a series of recording to one another. However, you cannot have nested recordings; the execution of the recording that contains the `%Aname` command stops after this command and is not resumed when the execution of `name` finishes. As a result, the data recorded after `%Aname` will never be played back. To avoid this, you should enter `%B` immediately after you have entered `%Aname` in a recording.

Interrupting a Playback

▶ To interrupt a recording that is played back in step mode

- Press CLEAR.

Once you have interrupted a recording, you have the following options:

- You can continue your session normally from the point where you stopped the recording.
- You can insert additional keyboard actions into the recording: after you have pressed CLEAR, enter the command **%B** and all actions you perform are inserted into the source of the recording until you enter **%B** again. Then, the execution of the recording is resumed.
- You can alter the next step in the recording: after you have pressed CLEAR, enter the command **%R**, then enter the input data for the next step. The newly entered input data overwrite the input data for this step in the recorded source. When you press ENTER, this step is executed with the new input data and subsequently the execution of the recording is resumed.
- You can execute any helproutine: after you have pressed CLEAR, enter the command **%J** directly followed by the name of the desired helproutine. The helproutine is invoked and the execution of the recording is continued as soon as the execution of the helproutine ends.

Manipulating a Recording

By recording the terminal command **%R**, you can manipulate a single step in a recording when it is played back. This applies in step mode and in background mode. In background mode, **%R** is the only way to interact with a recording that is being played back. Interaction, for example, may be required to provide an input option for sensitive data, such as passwords which are unknown at the time of the recording.

If the terminal command **%R** (redisplay last screen) has been recorded, the subsequent screen is open for user input when the recording is played back; that is, the input data for this screen are not taken from the recording but from what the user enters. Subsequently, the execution of the recording is continued.