NATURAL

Natural

Reference

Version 5.1.1 for Windows
Version 3.1.5 for Mainframes
Version 5.1.1 for UNIX and OpenVMS

software AG

This document applies to Natural Version 5.1.1 for Windows, Version 3.1.5 for Mainframes, Version 5.1.1 for UNIX and OpenVMS, 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.

© March 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

Reference - Overview 1
Reference - Overview 1
Parameter Reference 2
Parameter Reference 2
Profile Parameters 3
Profile Parameters 3
Alphabetical List of Profile Parameters 3
ACIPATT - Define Node Pattern for ACI Protocol 4
ACIPATT - Define Node Pattern for ACI Protocol 4
ACIVERS - Define ACI Version for Use with EntireX Broker ACI 5
ACIVERS - Define ACI Version for Use with EntireX Broker ACI 5
ACTPOLICY - Defines Default Activation Policy for DCOM Classes 6
ACTPOLICY - Defines Default Activation Policy for DCOM Classes 6
ADAMODE - Adabas Interface Mode 7
ADAMODE - Adabas Interface Mode 7
ADANAME - Name of Adabas Link Routine 9
ADANAME - Name of Adabas Link Routine 9
ADAPRM - REVIEW/DB Support 10
ADAPRM - REVIEW/DB Support 10
ADASBV - Adabas Security By Setting 11
ADASBV - Adabas Security By Setting 11
ASIZE - Entire System Server Auxiliary Buffer 12
ASIZE - Entire System Server Auxiliary Buffer 12
ASPSIZE (Internal Use) 13
ASPSIZE (Internal Use) 13
ASYNNAM - Output System ID for Asynchronous Processing 14
ASYNNAM - Output System ID for Asynchronous Processing 14
ATTN - Attention Key Interrupt Support 15
ATTN - Attention Key Interrupt Support 15
AUTO - Automatic Logon 16
AUTO - Automatic Logon 16
Mainframes 16
UNIX/OpenVMS and Windows 16
AUTOREGISTER - Automatic Registry Update 17
AUTOREGISTER - Automatic Registry Update 17
AUTORPC - Automatic Natural RPC Execution 18
AUTORPC - Automatic Natural RPC Execution 18
BATCH - Batch Mode Simulation 19
BATCH - Batch Mode Simulation 19
BATCHMODE - Real Batch Mode 20
BATCHMODE - Real Batch Mode 20
BMBLANK - Display Trailing Blanks 21
BMBLANK - Display Trailing Blanks 21
BMCONTROL - Display Control Characters 22
BMCONTROL - Display Control Characters 22
BMFRAME - Window Frame Characters 23
BMFRAME - Window Frame Characters 23
BMSIM - Similar Batch Mode Output 24
BMSIM - Similar Batch Mode Output 24
BMTIME - Display Process Time 25
BMTIME - Display Process Time 25
BMTITLE - Display Window Title 26
BMTITLE - Display Window Title 26

BMVERSION - Display Natural Version . . . 27
BMVERSION - Display Natural Version . . . 27
BPI - Buffer Pool Initialization . . . 28
BPI - Buffer Pool Initialization . . . 28
BPID - Specify Buffer Pool ID . . . 29
BPID - Specify Buffer Pool ID . . . 29
BPLIST - Name of Preload List for Natural Buffer Pool . . . 30
BPLIST - Name of Preload List for Natural Buffer Pool . . . 30
Mainframe-specific Information . . . 30
BPNAME - Name of Natural Global Buffer Pool . . . 31
BPNAME - Name of Natural Global Buffer Pool . . . 31
Mainframe-specific Information . . . 31
BPNLE - Number of Directory Entries . . . 32
BPNLE - Number of Directory Entries . . . 32
BPPROP - Global Buffer Pool Propagation . . . 33
BPPROP - Global Buffer Pool Propagation . . . 33
BPSFI - Object Search First in Buffer Pool . . . 34
BPSFI - Object Search First in Buffer Pool . . . 34
BPSIZE - Size of Natural Local Buffer Pool . . . 35
BPSIZE - Size of Natural Local Buffer Pool . . . 35
Mainframes . . . 35
UNIX/OpenVMS and Windows . . . 35
BPTEXT - Size of Text Segments in Natural Buffer Pool . . . 36
BPTEXT - Size of Text Segments in Natural Buffer Pool . . . 36
BSIZE - Size of EntireX Broker Buffer . . . 37
BSIZE - Size of EntireX Broker Buffer . . . 37
CANCEL - Session Cancellation with Dump . . . 38
CANCEL - Session Cancellation with Dump . . . 38
CC - Error Processing in Batch Mode . . . 39
CC - Error Processing in Batch Mode . . . 39
Mainframes . . . 39
UNIX/OpenVMS and Windows . . . 39
CCTAB - Printer Escape-Sequence Control Character . . . 40
CCTAB - Printer Escape-Sequence Control Character . . . 40
CDYNAM - Dynamic Loading of Non-Natural Programs . . . 41
CDYNAM - Dynamic Loading of Non-Natural Programs . . . 41
Mainframes . . . 41
UNIX/OpenVMS and Windows . . . 41
CF - Character for Terminal Commands . . . 42
CF - Character for Terminal Commands . . . 42
Mainframes . . . 42
UNIX/OpenVMS and Windows . . . 42
CFWSIZE (Internal Use) . . . 43
CFWSIZE (Internal Use) . . . 43
CLEAR - Processing of CLEAR Key in NEXT Mode . . . 44
CLEAR - Processing of CLEAR Key in NEXT Mode . . . 44
CM - Command Mode . . . 45
CM - Command Mode . . . 45
CMOBJIN - Batch Input File for Natural INPUT Data . . . 46
CMOBJIN - Batch Input File for Natural INPUT Data . . . 46
CMPO - Compilation Options . . . 47
CMPO - Compilation Options . . . 47
CMPRINT - Batch Output File . . . 48
CMPRINT - Batch Output File . . . 48
CMPRTnn - Additional Report . . . 49
CMPRTnn - Additional Report . . . 49

 Copyright © Software AG 2002

CMSYNIN - Batch Input File for Natural Commands and INPUT Data 50
CMSYNIN - Batch Input File for Natural Commands and INPUT Data 50
CMWRKnn - Natural Work Files 51
CMWRKnn - Natural Work Files 51
CO - Compiler Output 52
CO - Compiler Output 52
COMPR - Set RPC Buffer Compression 53
COMPR - Set RPC Buffer Compression 53
COMSERVERID - Determine DCOM Server ID 54
COMSERVERID - Determine DCOM Server ID 54
CP - Define Code Page Name 55
CP - Define Code Page Name 55
CSCPATT - Define RPC Client's Node Pattern 56
CSCPATT - Define RPC Client's Node Pattern 56
CSIZE - Size of Con-nect Buffer Area 57
CSIZE - Size of Con-nect Buffer Area 57
CSTATIC - Programs Statically Linked to Natural 58
CSTATIC - Programs Statically Linked to Natural 58
CVMIN - Control Variable Modified at Input 59
CVMIN - Control Variable Modified at Input 59
DATSIZE - Size of Buffer for Local Data 60
DATSIZE - Size of Buffer for Local Data 60
DB - Database Types and Options 62
DB - Database Types and Options 62
DB2SIZE - Natural Buffer Area for DB2 or SQL/DS 63
DB2SIZE - Natural Buffer Area for DB2 or SQL/DS 63
DBCLOSE - Database Close at Session End 64
DBCLOSE - Database Close at Session End 64
DBID - Default Database ID of Natural System Files 65
DBID - Default Database ID of Natural System Files 65
DBOPEN - Database Open Without ETID 66
DBOPEN - Database Open Without ETID 66
DBROLL - Database Calls Before Roll-Out 67
DBROLL - Database Calls Before Roll-Out 67
DBSHORT - Interpretation of Database Short Names 68
DBSHORT - Interpretation of Database Short Names 68
DBUPD - Database Updating 69
DBUPD - Database Updating 69
DC - Decimal Character 70
DC - Decimal Character 70
Mainframes 70
UNIX/OpenVMS and Windows 70
DCOM - NaturalX Support 71
DCOM - NaturalX Support 71
DD - Day Differential 72
DD - Day Differential 72
Mainframes 72
UNIX/OpenVMS and Windows 72
DELETE - Deletion of Dynamically Loaded Programs 73
DELETE - Deletion of Dynamically Loaded Programs 73
DFOUT - Date Format for Output 74
DFOUT - Date Format for Output 74
DFS - Specify RPC Client's Default Server Address 75
DFS - Specify RPC Client's Default Server Address 75
DFSTACK - Date Format for Stack 76
DFSTACK - Date Format for Stack 76

DFTITLE - Output Format of Date in Standard Report Title 77
DFTITLE - Output Format of Date in Standard Report Title 77
DLISIZE - Size of Natural Buffer Area for DL/I 78
DLISIZE - Size of Natural Buffer Area for DL/I 78
DSC - Data-Stream Compression (for 3270-Type Terminals) 79
DSC - Data-Stream Compression (for 3270-Type Terminals) 79
DSIZE - Size of Debug Buffer Area 80
DSIZE - Size of Debug Buffer Area 80
DSLM - Size Limitation 81
DSLM - Size Limitation 81
DTFORM - Date Format 82
DTFORM - Date Format 82
DU - Dump Generation 83
DU - Dump Generation 83
Mainframes 83
UNIX/OpenVMS and Windows 83
DYNPARM - Control Use of Dynamic Parameters 85
DYNPARM - Control Use of Dynamic Parameters 85
Mainframes 85
UNIX/OpenVMS and Windows 87
ECHO - Control Printing of Batch Input Data 88
ECHO - Control Printing of Batch Input Data 88
ECPMOD - Entire Connection Protocol Mode 89
ECPMOD - Entire Connection Protocol Mode 89
EDITOR - Other Program Editor 90
EDITOR - Other Program Editor 90
EDPSIZE - Size of Software AG Editor Auxiliary Buffer Pool 91
EDPSIZE - Size of Software AG Editor Auxiliary Buffer Pool 91
EDTBPSIZE - Software AG Editor Buffer Pool Size 92
EDTBPSIZE - Software AG Editor Buffer Pool Size 92
EDTLFILES - Number of Software AG Editor Logical Files 93
EDTLFILES - Number of Software AG Editor Logical Files 93
EDTRB - Program Editor Ring Buffer 94
EDTRB - Program Editor Ring Buffer 94
EJ - Page Eject 95
EJ - Page Eject 95
ENDBT - BACKOUT TRANSACTION at Session End 96
ENDBT - BACKOUT TRANSACTION at Session End 96
ENDMSG - Display Session-End Message 97
ENDMSG - Display Session-End Message 97
Mainframes 97
UNIX/OpenVMS and Windows 98
ENDIAN - Endian Mode for Compiled Objects 99
ENDIAN - Endian Mode for Compiled Objects 99
ESCAPE - Ignore Terminal Commands %% and %. 100
ESCAPE - Ignore Terminal Commands %% and %. 100
ESIZE - Size of User-Buffer Extension Area 101
ESIZE - Size of User-Buffer Extension Area 101
ESXDB - Database ID used for Entire System Server DDMs 102
ESXDB - Database ID used for Entire System Server DDMs 102
ET - Execution of END/BACKOUT TRANSACTION Statements 103
ET - Execution of END/BACKOUT TRANSACTION Statements 103
ETA - Error Transaction Program 104
ETA - Error Transaction Program 104
ETDB - Database for Transaction Data 106
ETDB - Database for Transaction Data 106

 Copyright © Software AG 2002

Mainframes 106
UNIX/OpenVMS and Windows 106
ETEOP - Issue END TRANSACTION at End of Program 107
ETEOP - Issue END TRANSACTION at End of Program 107
ETID - Adabas User Identification 108
ETID - Adabas User Identification 108
Mainframes 108
ETID - UNIX/OpenVMS and Windows 109
ETIO - Issue END TRANSACTION upon Terminal I/O 110
ETIO - Issue END TRANSACTION upon Terminal I/O 110
ETPDB - Databases Containing Entire Transaction Propagator Master Files 111
ETPDB - Databases Containing Entire Transaction Propagator Master Files 111
ETPSIZE - Size of Entire Transaction Propagator Buffer 112
ETPSIZE - Size of Entire Transaction Propagator Buffer 112
Mainframes 112
UNIX/OpenVMS and Windows 112
ETRACE - External Trace Function 113
ETRACE - External Trace Function 113
Using GTF 113
EXCSIZE - Size of Buffer for Natural Expert C Interface 114
EXCSIZE - Size of Buffer for Natural Expert C Interface 114
EXRSIZE - Size of Buffer for Natural Expert Rule Tables 115
EXRSIZE - Size of Buffer for Natural Expert Rule Tables 115
FAMSTD - Overwriting of Print and Work File Access Method Assignments 116
FAMSTD - Overwriting of Print and Work File Access Method Assignments 116
FC - Filler Character for INPUT Statement 117
FC - Filler Character for INPUT Statement 117
FCDP - Filler Character for Dynamically Protected Input Fields 118
FCDP - Filler Character for Dynamically Protected Input Fields 118
FDIC - Predict System File 119
FDIC - Predict System File 119
Mainframes 119
FDIC with VSAM System Files 119
UNIX/OpenVMS and Windows 120
FNAT - Natural System File for System Programs 121
FNAT - Natural System File for System Programs 121
Mainframes 121
FNAT with VSAM System Files 122
UNIX/OpenVMS and Windows 122
FNR - Default File Number of Natural System Files 123
FNR - Default File Number of Natural System Files 123
FS - Default Format/Length Setting for User-Defined Variables 124
FS - Default Format/Length Setting for User-Defined Variables 124
FSEC - Natural Security System File 125
FSEC - Natural Security System File 125
Mainframes 125
UNIX/OpenVMS and Windows 125
FSPOOL - Natural Advanced Facilities Spool File 127
FSPOOL - Natural Advanced Facilities Spool File 127
FSPOOL with VSAM System Files 127
FUSER - Natural System File for User Programs 128
FUSER - Natural System File for User Programs 128
Mainframes 128
FUSER with VSAM System Files 129
UNIX/OpenVMS and Windows 129

GFID - Global Format IDs 130
GFID - Global Format IDs 130
HCAM - Hardcopy Access Method 131
HCAM - Hardcopy Access Method 131
HCDEST - Hardcopy Output Destination 132
HCDEST - Hardcopy Output Destination 132
HI - Help Character 133
HI - Help Character 133
IA - Input Assign Character 134
IA - Input Assign Character 134
ID - Input Delimiter Character 135
ID - Input Delimiter Character 135
IDSIZE - Size of Buffer for Natural/IDMS 136
IDSIZE - Size of Buffer for Natural/IDMS 136
IKEY - Processing of PA and PF Keys 137
IKEY - Processing of PA and PF Keys 137
Mainframes 137
UNIX/OpenVMS and Windows 137
IM - Input Mode 138
IM - Input Mode 138
IMSG - Session Initialization Error Messages 139
IMSG - Session Initialization Error Messages 139
INIT-LIB - Library for Automatic Logon 140
INIT-LIB - Library for Automatic Logon 140
INTENS - Printing of Intensified Fields 141
INTENS - Printing of Intensified Fields 141
ISIZE - Size of Initialization Buffer 142
ISIZE - Size of Initialization Buffer 142
ITERM - Session Termination in Case of Initialization Error 143
ITERM - Session Termination in Case of Initialization Error 143
ITRACE - Internal Trace Function 144
ITRACE - Internal Trace Function 144
KAPRI - Kanji Printing 145
KAPRI - Kanji Printing 145
KC - Check for Statement Keywords 146
KC - Check for Statement Keywords 146
KEY - Setting Assignments to PA, PF and CLEAR Keys 147
KEY - Setting Assignments to PA, PF and CLEAR Keys 147
LC - Lower to Upper Case Translation 148
LC - Lower to Upper Case Translation 148
Mainframes 148
UNIX/OpenVMS and Windows 148
LDB - Wait Time for Response of Local Adabas Database 149
LDB - Wait Time for Response of Local Adabas Database 149
LE - Reaction when Limit for Processing Loop Exceeded 150
LE - Reaction when Limit for Processing Loop Exceeded 150
LFILE - Dynamic Specification of Logical File 151
LFILE - Dynamic Specification of Logical File 151
Mainframes 151
UNIX/OpenVMS and Windows 152
Password and Cipher Key 152
LFILMAX - Maximum Number of Dynamic Specifications of LFILE Parameter 153
LFILMAX - Maximum Number of Dynamic Specifications of LFILE Parameter 153
LIBNAM - Name of External Program Load Library 154
LIBNAM - Name of External Program Load Library 154

 Copyright © Software AG 2002

LOG (Internal Use) 155
LOG (Internal Use) 155
LOGONRQ - Logon for RPC Server Request Required 156
LOGONRQ - Logon for RPC Server Request Required 156
LS - Line Size 157
LS - Line Size 157
LSTEP - Initial Setting for *STEPLIB System Variable 158
LSTEP - Initial Setting for *STEPLIB System Variable 158
LT - Limit for Processing Loops 159
LT - Limit for Processing Loops 159
MADIO - Maximum DBMS Calls between Screen I/O Operations 160
MADIO - Maximum DBMS Calls between Screen I/O Operations 160
MAINPR - Override Default Output Report Number 161
MAINPR - Override Default Output Report Number 161
Mainframes 161
UNIX/OpenVMS and Windows 161
MAXBUFF - Maximum Buffer Size 162
MAXBUFF - Maximum Buffer Size 162
MAXCL - Maximum Number of Program Calls 163
MAXCL - Maximum Number of Program Calls 163
MAXROLL - Number of CMROLL Calls Before Roll-Out 164
MAXROLL - Number of CMROLL Calls Before Roll-Out 164
MAXUSER - Maximum Number of Users 165
MAXUSER - Maximum Number of Users 165
MENU - Menu Mode 166
MENU - Menu Mode 166
ML - Position of Message Line 167
ML - Position of Message Line 167
MONSIZE - Size of SYSTP Monitor Buffer 168
MONSIZE - Size of SYSTP Monitor Buffer 168
MP - Maximum Number of Pages of a Report 169
MP - Maximum Number of Pages of a Report 169
MSGSF - Display System Error Messages in Short/Full Format 170
MSGSF - Display System Error Messages in Short/Full Format 170
MT - Maximum CPU Time 171
MT - Maximum CPU Time 171
NAFSIZE - Size of Buffer for Natural Advanced Facilities 172
NAFSIZE - Size of Buffer for Natural Advanced Facilities 172
NAFUPF - Natural Advanced Facilities User Profile 173
NAFUPF - Natural Advanced Facilities User Profile 173
NATLOG - Natural Log File 174
NATLOG - Natural Log File 174
Platform-specific Examples 174
Example NATLOG output contents for OpenVMS: 174
Example NATLOG output contents for UNIX: 175
Example NATLOG output contents for Windows 176
NATVERS - Switching between Natural Environments 178
NATVERS - Switching between Natural Environments 178
NC - Use of Natural System Commands 179
NC - Use of Natural System Commands 179
NENTRY - Left/Right Alignment of Numeric Field Entries 180
NENTRY - Left/Right Alignment of Numeric Field Entries 180
NISN (Internal Use) 181
NISN (Internal Use) 181
NOAPPLERR - Suppress Message Number Prefix NAT 182
NOAPPLERR - Suppress Message Number Prefix NAT 182

NOPROX - Specify Local Domains for Direct Addressing . . . 183
NOPROX - Specify Local Domains for Direct Addressing . . . 183
NUCNAME - Name of Shared Nucleus . . . 184
NUCNAME - Name of Shared Nucleus . . . 184
OBJIN - Use of CMOBJIN as Natural Input File . . . 185
OBJIN - Use of CMOBJIN as Natural Input File . . . 185
OPF - Overwriting of Protected Fields by Helproutines . . . 186
OPF - Overwriting of Protected Fields by Helproutines . . . 186
OPRB - Database Open/Close Processing . . . 187
OPRB - Database Open/Close Processing . . . 187
Mainframes . . . 187
Dynamic OPRB with Natural Security . . . 187
OPRB for VSAM . . . 187
OPRB for Adabas . . . 188
UNIX/OpenVMS and Windows . . . 188
OPRB String Syntax . . . 189
OPT - Control of Natural Optimizer Compiler . . . 191
OPT - Control of Natural Optimizer Compiler . . . 191
OUTDEST - Output Destination for Asynchronous Processing . . . 192
OUTDEST - Output Destination for Asynchronous Processing . . . 192
PARM - Alternative Parameter Module . . . 193
PARM - Alternative Parameter Module . . . 193
Mainframes . . . 193
UNIX/OpenVMS and Windows . . . 193
PC - Control of Personal-Computer Access Method . . . 194
PC - Control of Personal-Computer Access Method . . . 194
Mainframe-specific Information . . . 194
PCNTRL - Print-Control Characters . . . 196
PCNTRL - Print-Control Characters . . . 196
PD - Number of Pages Recorded by NATPAGE . . . 197
PD - Number of Pages Recorded by NATPAGE . . . 197
PLOG - Logging of Dynamic Parameters . . . 198
PLOG - Logging of Dynamic Parameters . . . 198
PLUGIN - Enable the Natural Plug-In Components . . . 199
PLUGIN - Enable the Natural Plug-In Components . . . 199
PM - Print Mode . . . 200
PM - Print Mode . . . 200
PM - Mainframes . . . 200
PM - UNIX/OpenVMS and Windows . . . 200
POS22 - Version 2.2 Algorithm for POS System Function . . . 201
POS22 - Version 2.2 Algorithm for POS System Function . . . 201
PRGPAR - Data to be Passed to Program Receiving Control at Termination . . . 202
PRGPAR - Data to be Passed to Program Receiving Control at Termination . . . 202
PRINT - Printer Assignments . . . 203
PRINT - Printer Assignments . . . 203
PROFILE - Activate Dynamic Parameter Profile . . . 204
PROFILE - Activate Dynamic Parameter Profile . . . 204
PROGRAM - Non-Natural Program Receiving Control after Termination . . . 206
PROGRAM - Non-Natural Program Receiving Control after Termination . . . 206
Mainframes . . . 206
UNIX/OpenVMS and Windows . . . 206
PROX - Specify URL of Proxy Server . . . 208
PROX - Specify URL of Proxy Server . . . 208
PROXPORT - Specify Port Number of Proxy . . . 209
PROXPORT - Specify Port Number of Proxy . . . 209

 Copyright © Software AG 2002

PS - Page Size for Natural Reports 210
PS - Page Size for Natural Reports 210
PSEUDO - CICS Pseudo-Conversational Mode 211
PSEUDO - CICS Pseudo-Conversational Mode 211
RCA - Resolve Addresses of Static Non-Natural Programs 212
RCA - Resolve Addresses of Static Non-Natural Programs 212
RCALIAS - External Name Definition for Non-Natural Programs 213
RCALIAS - External Name Definition for Non-Natural Programs 213
RCFIND - Handling of Response Code 113 for FIND Statement 214
RCFIND - Handling of Response Code 113 for FIND Statement 214
RCGET - Handling of Response Code 113 for GET Statement 215
RCGET - Handling of Response Code 113 for GET Statement 215
RDACTIVE - Activate Remote Debugger 216
RDACTIVE - Activate Remote Debugger 216
RDCEXIT - Define Natural Data Collector User Exits 217
RDCEXIT - Define Natural Data Collector User Exits 217
RDCSIZE - Size of Buffer for the Natural Data Collector 218
RDCSIZE - Size of Buffer for the Natural Data Collector 218
RDNODE - Remote Debugger Node Name 219
RDNODE - Remote Debugger Node Name 219
RDPORT - Remote Debugger Port 220
RDPORT - Remote Debugger Port 220
RDS - Define Remote Directory Server 221
RDS - Define Remote Directory Server 221
READER - VSE/ESA System Logical Units for Input 222
READER - VSE/ESA System Logical Units for Input 222
RECAT - Dynamic Recataloging 223
RECAT - Dynamic Recataloging 223
Mainframes 223
UNIX/OpenVMS and Windows 223
REINP - Issue Internal REINPUT Statement for Invalid Data 225
REINP - Issue Internal REINPUT Statement for Invalid Data 225
RELO - Storage Thread Relocation 226
RELO - Storage Thread Relocation 226
RFILE - File for Recordings 227
RFILE - File for Recordings 227
RI - Release ISNs 228
RI - Release ISNs 228
RJESIZE - Initial Size of NATRJE Buffer 229
RJESIZE - Initial Size of NATRJE Buffer 229
RM - Retransmit Modified Fields 230
RM - Retransmit Modified Fields 230
ROSY - Read-Only Access to System Files 231
ROSY - Read-Only Access to System Files 231
RPC - Remote-Procedure-Call Settings 232
RPC - Remote-Procedure-Call Settings 232
RPCSIZE - Size of Buffer Used by Natural RPC 233
RPCSIZE - Size of Buffer Used by Natural RPC 233
RUNSIZE - Size of Runtime Buffer 234
RUNSIZE - Size of Runtime Buffer 234
SA - Sound Terminal Alarm 235
SA - Sound Terminal Alarm 235
SCTAB - Scanner Character-Type Table 236
SCTAB - Scanner Character-Type Table 236
SD - Time Delay between Two Screens 237
SD - Time Delay between Two Screens 237

SENDER - Screen Output Destination for Asynchronous Processing 238
SENDER - Screen Output Destination for Asynchronous Processing 238
SERVER - Start Natural Session as an RPC Server Session 240
SERVER - Start Natural Session as an RPC Server Session 240
SF - Spacing Factor 241
SF - Spacing Factor 241
SHELL - Grant Shell Access to Natural User 242
SHELL - Grant Shell Access to Natural User 242
SI - Shift-In Code for Double-Byte Character Set 243
SI - Shift-In Code for Double-Byte Character Set 243
SKEY - Storage Protection Key 244
SKEY - Storage Protection Key 244
SL - Source-Line Length 245
SL - Source-Line Length 245
SM - Programming in Structured Mode 246
SM - Programming in Structured Mode 246
SNAT - Sound a Bell at Syntax Error 247
SNAT - Sound a Bell at Syntax Error 247
SO - Shift-Out Code for Double-Byte Character Set 248
SO - Shift-Out Code for Double-Byte Character Set 248
SORT - Control of Sort Program 249
SORT - Control of Sort Program 249
SORTSZE - Size of Sort Buffer 250
SORTSZE - Size of Sort Buffer 250
SRVNAME - Name of RPC Server 251
SRVNAME - Name of RPC Server 251
SRVNODE - Name of Node 252
SRVNODE - Name of Node 252
SRVUSER - User ID for RPC Server Registry 253
SRVUSER - User ID for RPC Server Registry 253
SSIZE - Size of Source Area Allocated by the Editors 254
SSIZE - Size of Source Area Allocated by the Editors 254
Mainframes 254
UNIX/OpenVMS and Windows 254
STACK - Place Data/Commands on the Stack 255
STACK - Place Data/Commands on the Stack 255
Mainframes 255
UNIX/OpenVMS and Windows 256
STARTUP - Program Name for System Variable *STARTUP 258
STARTUP - Program Name for System Variable *STARTUP 258
STEPLIB - Default Steplib Library 259
STEPLIB - Default Steplib Library 259
Mainframes 259
UNIX/OpenVMS and Windows 259
SUBSID - Subsystem ID under OS/390 and VSE/ESA 260
SUBSID - Subsystem ID under OS/390 and VSE/ESA 260
SYMGEN - Generate Symbol Table 261
SYMGEN - Generate Symbol Table 261
SYNERR - Control of Syntax Errors 262
SYNERR - Control of Syntax Errors 262
SYS - Activate Set of Dynamic Profile Parameters 263
SYS - Activate Set of Dynamic Profile Parameters 263
SYSCIP - Adabas Cipher Key for Natural System Files 264
SYSCIP - Adabas Cipher Key for Natural System Files 264
SYSPSW - Adabas Password for Natural System Files 265
SYSPSW - Adabas Password for Natural System Files 265

 Copyright © Software AG 2002

TAB - Standard Output Translation Table 266
TAB - Standard Output Translation Table 266
TAB1 - Alternative Output Translation Table 267
TAB1 - Alternative Output Translation Table 267
TAB2 - Alternative Input Translation Table 268
TAB2 - Alternative Input Translation Table 268
TABA1 - EBCDIC-to-ASCII Translation Table 269
TABA1 - EBCDIC-to-ASCII Translation Table 269
TABA2 - ASCII-to-EBCDIC Translation Table 270
TABA2 - ASCII-to-EBCDIC Translation Table 270
TABL - Translation Table for Output from "SYS" Libraries 271
TABL - Translation Table for Output from "SYS" Libraries 271
TD - Time Differential 272
TD - Time Differential 272
Mainframes 272
UNIX/OpenVMS and Windows 272
TF - Translation of Database ID/File Number 274
TF - Translation of Database ID/File Number 274
Mainframes 274
UNIX/OpenVMS and Windows 275
TIMEOUT - Wait Time for RPC Server Response 276
TIMEOUT - Wait Time for RPC Server Response 276
TMPSORTUNIQ - Alternate Algorithm for Generating Sort Work File Names 277
TMPSORTUNIQ - Alternate Algorithm for Generating Sort Work File Names 277
TPF (Internal Use) 278
TPF (Internal Use) 278
TQ - Translate Quotation Marks 279
TQ - Translate Quotation Marks 279
TRACE - Define Trace Level for Natural RPC Servers 280
TRACE - Define Trace Level for Natural RPC Servers 280
TRACE - Define Mainframe Components to be Traced 281
TRACE - Define Mainframe Components to be Traced 281
TRANSP - Server Transport Protocol 282
TRANSP - Server Transport Protocol 282
TRYALT - Try Alternative Server Address 283
TRYALT - Try Alternative Server Address 283
TS - Translate Output from Programs in System Libraries 284
TS - Translate Output from Programs in System Libraries 284
TSIZE - Size of Buffer for Adabas Text Retrieval 285
TSIZE - Size of Buffer for Adabas Text Retrieval 285
TTYPE - Terminal Type 286
TTYPE - Terminal Type 286
UDB - User Database ID 287
UDB - User Database ID 287
Mainframes 287
UNIX/OpenVMS and Windows 287
ULANG - User Language 288
ULANG - User Language 288
USEDIC - Common Logical Name for Dictionary Servers 289
USEDIC - Common Logical Name for Dictionary Servers 289
USER - Restrict the Use of Profile Parameter Strings 290
USER - Restrict the Use of Profile Parameter Strings 290
Mainframes 290
UNIX/OpenVMS and Windows 290
USERBUF (Internal Use) 292
USERBUF (Internal Use) 292

USEREP - Repository Usage . . . 293
USEREP - Repository Usage . . . 293
USIZE - Size of User Buffer . . . 294
USIZE - Size of User Buffer . . . 294
UTAB1 - Translation Table for Lower to Upper Case . . . 295
UTAB1 - Translation Table for Lower to Upper Case . . . 295
UTAB2 - Translation Table for Upper to Lower Case . . . 296
UTAB2 - Translation Table for Upper to Lower Case . . . 296
V22COMP - Compatibility with Inconsistent Version 2.2 Syntax . . . 297
V22COMP - Compatibility with Inconsistent Version 2.2 Syntax . . . 297
VSIZE - Size of Buffer Area for Natural/VSAM . . . 298
VSIZE - Size of Buffer Area for Natural/VSAM . . . 298
WFOPFA - Opening of Work Files . . . 299
WFOPFA - Opening of Work Files . . . 299
WH - Wait for Record in Hold Status . . . 300
WH - Wait for Record in Hold Status . . . 300
WORK - Work-File Assignments . . . 301
WORK - Work-File Assignments . . . 301
Mainframes . . . 301
UNIX/OpenVMS and Windows . . . 301
WPSIZE - Sizes of Natural Work Pools . . . 302
WPSIZE - Sizes of Natural Work Pools . . . 302
WSISIZE - Buffer for Natural Workstation Interface . . . 303
WSISIZE - Buffer for Natural Workstation Interface . . . 303
WSIZE - Size of Buffer Area for Screen Images . . . 304
WSIZE - Size of Buffer Area for Screen Images . . . 304
XADB - Database for Coordination of Transaction Processing . . . 305
XADB - Database for Coordination of Transaction Processing . . . 305
XREF - Active Cross-Reference Feature . . . 306
XREF - Active Cross-Reference Feature . . . 306
Mainframes . . . 306
UNIX/OpenVMS and Windows . . . 306
XSIZE - Size of Buffer for User Subsystem . . . 307
XSIZE - Size of Buffer for User Subsystem . . . 307
YD - Year Differential . . . 308
YD - Year Differential . . . 308
YSLW - Year Sliding Window . . . 309
YSLW - Year Sliding Window . . . 309
ZD - Zero-Division Check . . . 310
ZD - Zero-Division Check . . . 310
ZP - Zero Printing . . . 311
ZP - Zero Printing . . . 311
Mainframes . . . 311
ZP - Zero Printing for UNIX/OpenVMS and Windows . . . 311
ZSIZE - Size of Entire DB Buffer Area . . . 312
ZSIZE - Size of Entire DB Buffer Area . . . 312
Session Parameters . . . 313
Session Parameters . . . 313
General Information . . . 313
How to Set Parameters . . . 313
Changing Parameters at Session Level - The GLOBALS Command . . . 313
Changing Parameters at Program Level - The FORMAT Statement . . . 314
Changing Parameters at Statement Level . . . 314
Changing Parameters at Field Level . . . 315
Evaluation of Parameters . . . 316
Summary of Parameters . . . 316

 Copyright © Software AG 2002

AD - Attribute Definition 321
AD - Attribute Definition 321
Field Representation 321
Field Alignment 322
Field Input/Output Characteristics 322
Mandatory Input 322
Length of Input Value 323
Field Upper/Lower Case Characteristics 323
Filler Character 323
AL - Alphanumeric Length for Output 324
AL - Alphanumeric Length for Output 324
BX - Box Definition 325
BX - Box Definition 325
CC - Conditional Program Execution 326
CC - Conditional Program Execution 326
CD - Color Definition 327
CD - Color Definition 327
CF - Character for Terminal Commands 328
CF - Character for Terminal Commands 328
CO - Compiler Output 329
CO - Compiler Output 329
CV - Control Variable 330
CV - Control Variable 330
DC - Character for Decimal Point Notation 331
DC - Character for Decimal Point Notation 331
DF - Date Format 332
DF - Date Format 332
DFOUT - Date Format for Output 333
DFOUT - Date Format for Output 333
DFSTACK - Date Format for Stack 334
DFSTACK - Date Format for Stack 334
DFTITLE - Date Format in Default Page Title 335
DFTITLE - Date Format in Default Page Title 335
DU - Dump Generation 336
DU - Dump Generation 336
DU for Mainframes 336
DU for UNIX/OpenVMS and Windows 337
DY - Dynamic Attributes 338
DY - Dynamic Attributes 338
EJ - Page Eject 340
EJ - Page Eject 340
EM - Edit Mask 341
EM - Edit Mask 341
Blanks in Edit Masks 341
Default Edit Masks 342
Edit Masks for Numeric Fields 342
Characters for the Definition of Numeric Edit Masks: 343
Edit Masks for Alphanumeric Fields 345
Hexadecimal Edit Masks 346
Edit Masks for Date and Time Fields - Formats D and T 347
Date - format D, and Time - format T: 347
Time - format T - only: 348
Examples of Date and Time Edit Masks: 348
Edit Masks for Logical Fields - Format L 349
Example of Edit Masks for Logical Field: 349

ENDIAN - Endian Mode for Compiled Objects . . . 350
ENDIAN - Endian Mode for Compiled Objects . . . 350
ES - Empty Line Suppression . . . 351
ES - Empty Line Suppression . . . 351
FC - Filler Character . . . 352
FC - Filler Character . . . 352
FCDP - Filler Character for Dynamically Protected Fields . . . 353
FCDP - Filler Character for Dynamically Protected Fields . . . 353
FL - Floating Point Mantissa Length . . . 354
FL - Floating Point Mantissa Length . . . 354
FS - Format Specification . . . 355
FS - Format Specification . . . 355
GC - Filler Character for Group Headers . . . 356
GC - Filler Character for Group Headers . . . 356
HC - Header Centering . . . 357
HC - Header Centering . . . 357
HE - Helproutine . . . 358
HE - Helproutine . . . 358
Execution Of Helproutines . . . 359
HW - Heading Width . . . 360
HW - Heading Width . . . 360
IA - INPUT Assign Character . . . 361
IA - INPUT Assign Character . . . 361
IC - Insertion Character . . . 362
IC - Insertion Character . . . 362
ID - INPUT Delimiter Character . . . 363
ID - INPUT Delimiter Character . . . 363
IM - INPUT Mode . . . 364
IM - INPUT Mode . . . 364
IP - INPUT Prompting Text . . . 365
IP - INPUT Prompting Text . . . 365
IS - Identical Suppress . . . 366
IS - Identical Suppress . . . 366
KD - Key Definition . . . 367
KD - Key Definition . . . 367
LC - Leading Characters . . . 368
LC - Leading Characters . . . 368
LE - Limit Error Processing . . . 369
LE - Limit Error Processing . . . 369
LS - Line Size . . . 370
LS - Line Size . . . 370
LT - Limit of Records Read . . . 371
LT - Limit of Records Read . . . 371
MC - Multiple-Value Field Count . . . 372
MC - Multiple-Value Field Count . . . 372
ML - Msg Line Position . . . 373
ML - Msg Line Position . . . 373
MP - Maximum Number of Pages . . . 374
MP - Maximum Number of Pages . . . 374
MS - Manual Skip . . . 375
MS - Manual Skip . . . 375
MT - Maximum CPU Time . . . 376
MT - Maximum CPU Time . . . 376
NC - Use of Natural System Commands . . . 377
NC - Use of Natural System Commands . . . 377

 Copyright © Software AG 2002

NL - Numeric Length for Output . . . 378
NL - Numeric Length for Output . . . 378
OPF - Overwriting of Protected Fields by Helproutines . . . 379
OPF - Overwriting of Protected Fields by Helproutines . . . 379
PC - Periodic Group Count . . . 380
PC - Periodic Group Count . . . 380
PD - NATPAGE Page Dataset . . . 381
PD - NATPAGE Page Dataset . . . 381
PM - Print Mode . . . 382
PM - Print Mode . . . 382
PS - Page Size . . . 383
PS - Page Size . . . 383
REINP - Internal REINPUT for Invalid Data . . . 384
REINP - Internal REINPUT for Invalid Data . . . 384
SA - Sound Terminal Alarm . . . 385
SA - Sound Terminal Alarm . . . 385
SF - Spacing Factor . . . 386
SF - Spacing Factor . . . 386
SG - Sign Position . . . 387
SG - Sign Position . . . 387
SL - Source Line Length . . . 388
SL - Source Line Length . . . 388
SM - Structured Mode . . . 389
SM - Structured Mode . . . 389
SYMGEN - Generate Symbol Table . . . 390
SYMGEN - Generate Symbol Table . . . 390
TC - Trailing Characters . . . 391
TC - Trailing Characters . . . 391
TS - Translate Output from Programs in System Libraries . . . 392
TS - Translate Output from Programs in System Libraries . . . 392
UC - Underlining Character . . . 393
UC - Underlining Character . . . 393
WH - Wait for Record in Hold Status . . . 394
WH - Wait for Record in Hold Status . . . 394
ZD - Zero Division Check . . . 395
ZD - Zero Division Check . . . 395
ZP - Zero Printing . . . 396
ZP - Zero Printing . . . 396
Parameter Modules - Overview . . . 397
Parameter Modules - Overview . . . 397
NTALIAS Macro - External Names of Non-Natural Programs . . . 399
NTALIAS Macro - External Names of Non-Natural Programs . . . 399
NTALIAS Syntax . . . 399
Example of NTALIAS Macro . . . 399
NTBPI Macro - Buffer Pool Initialization . . . 400
NTBPI Macro - Buffer Pool Initialization . . . 400
NTBPI Syntax . . . 400
NTBPI Parameters . . . 400
TYPE - Type of Buffer Pool . . . 400
SEQ - Sequence Number of Buffer Pool . . . 400
NAME - Name of Global Buffer Pool . . . 401
SIZE - Size of Buffer Pool . . . 401
LIST - Name of Preload List to be Used . . . 401
TXTSIZE - Size of Buffer Pool Text Segments . . . 401
Examples of NTBPI Macros . . . 402
NTBPI Alternative . . . 402

NTCCTAB Macro - Printer Escape-Character Control Table 403
NTCCTAB Macro - Printer Escape-Character Control Table 403
NTCCTAB Syntax 403
1. Variant 403
2. Variant 403
3. Variant 404
String Syntax for CODE, CS, CCS or CSE 404
Proportional Fonts 404
Example of NTCCTAB Macro 404
NTCMPO Macro - Compilation Options 405
NTCMPO Macro - Compilation Options 405
NTCMPO Syntax 405
Keyword Parameters for NTCMPO Macro 405
Example of NTCMPO Macro 405
NTCMPO Alternative 405
NTDB Macro - Database Types and Options 407
NTDB Macro - Database Types and Options 407
NTDB Syntax 407
NTDB Options 408
Examples of NTDB Macros 408
Example 1 of NTDB Macros 408
Example 2 of NTDB Macros 408
NTDB Alternative 409
NTDCOM - NaturalX Support 410
NTDCOM - NaturalX Support 410
NTDCOM Syntax 410
NTDCOM Parameters 410
ACTPOL - Default Activation Policy for NaturalX Classes 410
AUTOREG - Register NaturalX Classes when Cataloging 410
SERVID - Server Name Used by NaturalX 410
Example of NTDCOM Alias 411
NTDCOM Alternative 411
NTDYNP Macro - Control Use of Dynamic Parameters 412
NTDYNP Macro - Control Use of Dynamic Parameters 412
NTDYNP Syntax 412
1. Form 412
2. Form 412
3. Form 412
Example of NTDYNP Macros 413
NTDYNP Alternative 413
NTFILE Macro - Logical Files 414
NTFILE Macro - Logical Files 414
NTFILE Syntax 414
NTFILE Parameters 414
Example of NTFILE Macro 415
NTFILE Alternative 415
NTKAPRI Macro - Kanji Printing 416
NTKAPRI Macro - Kanji Printing 416
NTKAPRI Syntax 416
NTKAPRI Alternative 416
NTOPRB Macro - Database Open Processing 417
NTOPRB Macro - Database Open Processing 417
NTOPRB Syntax 417
Examples of NTOPRB Macros 417
NTOPRB Alternative 417

 Copyright © Software AG 2002

NTOPT Macro - Control of Natural Optimizer Compiler 418
NTOPT Macro - Control of Natural Optimizer Compiler 418
NTPRINT Macro - Print File Assignments 419
NTPRINT Macro - Print File Assignments 419
NTPRINT Syntax 419
print-file-numbers 419
keyword-parameters 420
NTPRINT Keyword Parameters for All Environments 420
AM - Type of Access Method 420
DEST - External Dataset Name 420
OPEN - Time of File Opening 421
CLOSE - Time of File Closure 421
NTPRINT Keyword Parameters for AM=STD in All Environments 422
RECFM - Default Record Format of Dataset 422
BLKSIZE - Default Block Size of Dataset 422
LRECL - Default Record Length of Dataset 422
TRUNC - Truncation of Output Records 423
PAD - Padding of output records 423
NTPRINT Keyword Parameters for AM=STD in OS/390 Environments 423
REREAD - Option for Closing of Tape File 423
FREE - Dataset De-allocation at File Closure 423
ASA - Use of ASA Record Format 424
BUFNO - Default Number of OS/390 I/O Buffers of Dataset 424
DISP - Open Print File for Modification 424
VMAX - Enforce LRECL=BLKSIZE-4 for Variable Record Format 424
NTPRINT Keyword Parameters for AM=STD in VSE/ESA Environments 424
SYSNR - Logical VSE SYS Number 424
LABEL - Tape Label Processing 425
REWIND - Action at File Closure 425
NTPRINT Keyword Parameters for AM=STD in BS2000/OSD Environments 425
DISP - File Open Mode 425
NTPRINT Keyword Parameters for AM=COMP 425
DRIVER - Name of Com-plete Print Driver 425
NTPRINT Keyword Parameters for AM=IMS 425
DRIVER - Name of Natural IMS Print Driver 426
NTPRINT Keyword Parameters for DEFINE PRINTER Statement 426
PROFILE - Name of Printer Control Characters Table 426
NAME - Name of Listing 426
FORMS - Name of Listing Forms 426
DISP - Listing Disposition 426
COPIES - Number of Copies 426
CLASS - Spool Class 426
PRTY - Listing Priority 426
NTPRINT Alternative 426
NTPRM Macro - Create Natural Parameter Module 428
NTPRM Macro - Create Natural Parameter Module 428
NTPRM Syntax 428
Assembler Macro Coding Conventions 428
NTRPC Macro - Handling of Natural Remote Procedure Calls 429
NTRPC Macro - Handling of Natural Remote Procedure Calls 429
NTRPC Syntax 429
NTRPC Macro Keyword Parameters that Apply to Client and Server 429
ACIVERS - Define ACI Version for Use with EntireX Broker ACI 430
MAXBUFF - Size of Buffers 430
RPCSIZE - Size of Buffer Used by Natural RPC 430
SERVER - Start Natural Session as an RPC Server Session 431

NTRPC Macro Keyword Parameters that Apply to the Server Only . . . 431
LOGONRQ - Logon for RPC Server Request Required . . . 431
NTASKS - Number of Server Tasks to be Started . . . 431
SRVNAME - Name of RPC Server . . . 432
SRVNODE - Name of Node . . . 432
SRVUSER - Server User ID for RPC Server Registry . . . 432
TRACE - Activate RPC Trace Facility . . . 432
TRANSP - Server Transport Protocol . . . 433
NTRPC Macro Keyword Parameters that Apply to the Client Only . . . 433
ACIPATT - Define Node Pattern for ACI Protocol . . . 433
AUTORPC - Automatic Natural RPC Execution . . . 434
COMPR - Set RPC Buffer Compression . . . 434
DFS - RPC Default Server Address . . . 435
RDS - Define Remote Directory Servers . . . 435
TIMEOUT - Wait Time for RPC Server Response . . . 436
TRYALT - Try Alternative Server . . . 437
NTSCTAB Macro - Overwriting of Scanner Character-Type Table . . . 438
NTSCTAB Macro - Overwriting of Scanner Character-Type Table . . . 438
NTSCTAB Syntax . . . 438
Example of NTSCTAB Macro . . . 438
NTSCTAB Alternative . . . 439
NTSORT Macro - Control of Sort Program . . . 440
NTSORT Macro - Control of Sort Program . . . 440
NTSORT Syntax . . . 440
NTSORT Keyword Parameters . . . 440
WRKSIZE - Size of Work Buffer Used by Sort Program . . . 440
STORAGE - Type of Storage Medium . . . 440
EXT - Use of External Sort Program . . . 441
EXTNAME - Name of External Sort Program . . . 441
EXTOPT - Additional Options for External Sort Program . . . 441
EXTEOJ - Action in the Event of an Error . . . 442
NTSORT Alternative . . . 442
NTSYS Macro - Specification of Set of Dynamic Parameters . . . 443
NTSYS Macro - Specification of Set of Dynamic Parameters . . . 443
NTSYS Syntax . . . 443
set-name . . . 443
parameter . . . 443
Passing the Parameter Set to Natural . . . 443
Macro Coding Conventions . . . 443
Examples of NTSYS Macros . . . 444
NTTAB Macro - Overwriting of Standard Output Translation Table . . . 445
NTTAB Macro - Overwriting of Standard Output Translation Table . . . 445
NTTAB Syntax . . . 445
Example of NTTAB Macro . . . 445
NTTAB Alternative . . . 445
NTTAB1 Macro - Overwriting of Alternate Output Translation Table . . . 446
NTTAB1 Macro - Overwriting of Alternate Output Translation Table . . . 446
NTTAB1 Syntax . . . 446
NTTAB1 Alternative . . . 446
NTTAB2 Macro - Overwriting of Alternate Input Translation Table . . . 447
NTTAB2 Macro - Overwriting of Alternate Input Translation Table . . . 447
NTTAB2 Syntax . . . 447
NTTAB2 Alternative . . . 447
NTTABA1 Macro - Overwriting of EBCDIC-ASCII Translation Table . . . 448
NTTABA1 Macro - Overwriting of EBCDIC-ASCII Translation Table . . . 448
NTTABA1 Syntax . . . 448

 Copyright © Software AG 2002

NTTABA1 Alternative 448
NTTABA2 Macro - Overwriting of ASCII-EBCDIC Translation Table 449
NTTABA2 Macro - Overwriting of ASCII-EBCDIC Translation Table 449
NTTABA2 Syntax 449
NTTABA2 Alternative 449
NTTABL Macro - Overwriting of SYS-Library Output Translation Table 450
NTTABL Macro - Overwriting of SYS-Library Output Translation Table 450
NTTABL Syntax 450
NTTABL Alternative 450
NTTF Macro - Translation of Database ID / File Number 451
NTTF Macro - Translation of Database ID / File Number 451
NTTF Syntax 451
Examples of NTTF Macros 452
NTTRACE Macro - Define Components for Tracing 453
NTTRACE Macro - Define Components for Tracing 453
NTTRACE Syntax 453
trace-ID 453
Example of NTTRACE Macro 453
NTTRACE Alternative 453
NTUSER Macro - Use of Alternative Parameter Module 454
NTUSER Macro - Use of Alternative Parameter Module 454
NTUSER Syntax 454
user-id 454
Example of NTUSER Macro 454
NTUSER Alternative 454
NTUTAB1 Macro - Overwriting of Lower-to-Upper-Case Translation Table 455
NTUTAB1 Macro - Overwriting of Lower-to-Upper-Case Translation Table 455
NTUTAB1 Syntax 455
NTUTAB1 Alternative 455
NTUTAB2 Macro - Overwriting of Upper-to-Lower-Case Translation Table 456
NTUTAB2 Macro - Overwriting of Upper-to-Lower-Case Translation Table 456
NTUTAB2 Syntax 456
NTUTAB2 Alternative 456
NTWORK Macro - Work File Assignments 457
NTWORK Macro - Work File Assignments 457
NTWORK Syntax 457
work-file-numbers 457
keyword-parameters 457
NTWORK Keyword Parameters for All Environments 458
AM - Type of Access Method 458
DEST - External Dataset Name 458
OPEN - Time of File Opening 459
CLOSE - Time of File Closure 459
NTWORK Keyword Parameters for AM=STD in All Environments 460
RECFM - Default Record Format of Dataset 460
BLKSIZE - Default Block Size of Dataset 460
LRECL - Default Record Length of Dataset 460
TRUNC - Truncation of Output Records 461
PAD - Padding of output records 461
NTWORK Keyword Parameters for AM=STD in OS/390 Environments 461
REREAD - Closing of Tape File 461
FREE - Dataset De-allocation at File Closure 461
BUFNO - Default Number of OS/390 I/O Buffers of Dataset 462
DISP - Open Work File for Modification 462
VMAX - Enforce LRECL=BLKSIZE-4 for Variable Record Format 462
NTWORK Keyword Parameters for AM=STD in VSE/ESA Environments 462

SYSNR - Logical VSE SYS Number . . . 462
LABEL - Tape Label Processing . . . 462
REWIND - Action at File Closure . . . 463
NTWORK Keyword Parameters for AM=STD in BS2000/OSD Environments . . . 463
DISP - File Open Mode . . . 463
NTWORK Keyword Parameters for AM=CICS . . . 463
TYPE - Type of CICS Storage Medium . . . 463
DISP - CICS Storage Queue Disposition . . . 463
NTWORK Keyword Parameters for AM=COMP . . . 464
TYPE - Type of Storage Access . . . 464
BLOCKS - Number of Storage Blocks . . . 464
BLKSIZE - Size of Storage Blocks . . . 464
NTWORK Alternative . . . 464
Command Reference - Overview . . . 465
Command Reference - Overview . . . 465
What are System Commands? . . . 465
Issuing System Commands . . . 465
Command Line . . . 466
MORE Prompt . . . 466
Syntax . . . 467
Syntax . . . 467
Commands Grouped by Functions . . . 469
Commands Grouped by Functions . . . 469
Navigating within Natural . . . 469
Environment Settings . . . 470
Editing and Storing Programming Objects . . . 471
Executing Programs . . . 472
Invoking Maintenance Utilities . . . 472
Invoking Transfer Utilities . . . 472
Monitoring and Debugging . . . 473
Commands used with NaturalX . . . 473
Commands used with SQL Databases . . . 473
Commands used with the Natural Optimizer Compiler . . . 474
Miscellaneous . . . 474
System Command List . . . 476
System Command List . . . 476
AIV . . . 477
AIV . . . 477
BUS . . . 478
BUS . . . 478
CATALL . . . 479
CATALL . . . 479
CATALL for Mainframes . . . 479
Catalog Objects from/to . . . 479
Recatalog Only Existing Modules, or Catalog All Sources . . . 480
Select Object Types . . . 480
Select Function . . . 480
Select Options . . . 480
Selection List . . . 481
Direct Command Syntax . . . 481
CATALL for UNIX/OpenVMS . . . 484
CATALL for Windows . . . 486
CATALOG . . . 488
CATALOG . . . 488
CATALOG for Mainframes . . . 488
CATALOG for UNIX/OpenVMS . . . 489

 Copyright © Software AG 2002

CATALOG for Windows . . . 490
CHECK . . . 491
CHECK . . . 491
CHECK for Mainframes . . . 491
CHECK for UNIX/OpenVMS and Windows . . . 492
CLEAR . . . 493
CLEAR . . . 493
CLEAR for Mainframes . . . 493
CLEAR for UNIX/OpenVMS and Windows . . . 494
COMPOPT . . . 495
COMPOPT . . . 495
COMPOPT for Mainframes . . . 495
General Information on Compiler Options . . . 495
Options . . . 496
KCHECK - Keyword Checking . . . 496
PCHECK - Parameter Checking for CALLNAT Statements . . . 496
DBSHORT - Interpretation of Database Short Field Names . . . 496
PSIGNF - Internal Representation of Positive Sign of Packed Numbers . . . 497
TSENABL - Applicability of TS Profile Parameter . . . 497
GFID - Generation of Global Format IDs . . . 498
LOWSRCE - Allow Lower-Case Source . . . 499
FINDMUN - Detect Inconsistent Comparison Logic in FIND Statements . . . 500
V22COMP - Compatibility Option for Inconsistent Version 2.2 Syntax - warning still valid??? . 501
COMPOPT for UNIX/OpenVMS . . . 503
COMPOPT for Windows . . . 505
Interpretation of Database Short Field Names - DBSHORT: . . . 505
Compatibility Option for Inconsistent Version 2.2 Syntax - V22COMP: . . . 505
Generation of Global Format IDs - GFID: . . . 505
DCOMPARM . . . 507
DCOMPARM . . . 507
DEBUG . . . 508
DEBUG . . . 508
DELETE . . . 509
DELETE . . . 509
object-name . . . 509
object-type . . . 510
Selection List . . . 510
Safeguard Against Accidental Deletion . . . 510
Examples . . . 511
DUMP . . . 512
DUMP . . . 512
EDIT . . . 513
EDIT . . . 513
EDIT for Mainframes . . . 513
object-type . . . 513
object-name . . . 514
library-id . . . 514
EDIT for UNIX/OpenVMS . . . 516
EDIT for Windows . . . 518
Edit Function . . . 519
EDT . . . 520
EDT . . . 520
Syntax Explanation . . . 520
object-name . . . 520
library-id . . . 520
EDT Subcommands . . . 521

EDT Function Keys . . . 522
EXECUTE . . . 523
EXECUTE . . . 523
EXECUTE for Mainframes . . . 523
Syntax Explanation . . . 523
REPEAT . . . 523
program-name . . . 523
library-id . . . 523
Passing Parameters to the Program . . . 524
EXECUTE for UNIX/OpenVMS and Windows . . . 525
Passing Parameters to the Program . . . 525
FIN . . . 527
FIN . . . 527
GLOBALS . . . 528
GLOBALS . . . 528
HELP . . . 529
HELP . . . 529
HELP for Mainframes . . . 529
HELP for UNIX/OpenVMS and Windows . . . 530
INPL . . . 531
INPL . . . 531
INPL for Mainframes . . . 531
INPL R - Natural Security Recover . . . 531
INPL for UNIX/OpenVMS . . . 532
INPL R - Natural Security Recover . . . 532
INPL for Windows . . . 533
INPL R - Natural Security Recover . . . 533
KEY . . . 534
KEY . . . 534
KEY for Mainframes . . . 534
Natural Utilities for Mainframes Assigning Commands . . . 534
Activating/Deactivating All Keys - KEY ON/OFF . . . 535
Activating/Deactivating Individual Keys - KEY *key*=ON/OFF . . . 535
KEY for UNIX/OpenVMS . . . 536
Assigning Commands . . . 536
Activating/Deactivating All Keys - KEY ON/OFF . . . 537
Activating/Deactivating Individual Keys - KEY *key*=ON/OFF . . . 537
LAST . . . 538
LAST . . . 538
LASTMSG . . . 540
LASTMSG . . . 540
LIST . . . 541
LIST . . . 541
LIST for Mainframes . . . 541
Syntax Diagrams . . . 542
object-type . . . 542
extended-type . . . 542
options . . . 543
Functions . . . 544
Displaying an Individual Source . . . 544
Displaying Directory Information . . . 544
Displaying Long Names of Cataloged Subroutines and Classes . . . 544
LIST SEQUENTIAL . . . 545
Displaying DDMs . . . 545
Displaying XREF Data . . . 545
LIST EXT extended-type object-name and LIST object-type object-name . . . 546

 Copyright © Software AG 2002

Options . . . 547
NUMBERS OFF . . . 547
FORMATTED . . . 547
SETTINGS . . . 547
FIELDS . . . 547
EXTFIELDS . . . 547
RULES/INLINERULES/FREERULES/AUTORULES . . . 547
List of Objects . . . 548
Scrolling the Selection List of Objects . . . 548
New Criteria for the Selection List . . . 548
Information Displayed on the Selection List . . . 548
Items Intensified on the Selection List . . . 548
Subcommands for a List of Objects . . . 549
Selecting an Object from the Selection List . . . 550
List of Source . . . 551
Subcommands for Listed Source . . . 551
FORMAT . . . 553
Cursor-Sensitive Object Selection . . . 554
expand-option . . . 555
Subcommands in Expand Object . . . 556
LIST for UNIX/OpenVMS . . . 557
Displaying an Individual Source . . . 557
Displaying Directory Information . . . 558
LIST *object-type object-name* . . . 558
LIST for Windows . . . 560
LIST COUNT . . . 562
LIST COUNT . . . 562
LIST DIR . . . 563
LIST DIR . . . 563
LIST XREF . . . 564
LIST XREF . . . 564
LISTSQL . . . 565
LISTSQL . . . 565
LOGOFF . . . 566
LOGOFF . . . 566
LOGON . . . 567
LOGON . . . 567
Naming Conventions for Libraries . . . 567
MAIL . . . 568
MAIL . . . 568
MAINMENU . . . 569
MAINMENU . . . 569
MAP . . . 570
MAP . . . 570
NATLOAD . . . 571
NATLOAD . . . 571
NATUNLD . . . 572
NATUNLD . . . 572
NOCOPT . . . 573
NOCOPT . . . 573
NOCSHOW . . . 574
NOCSHOW . . . 574
NOCSTAT . . . 575
NOCSTAT . . . 575
PROFILE . . . 576
PROFILE . . . 576

PURGE . . . 577
PURGE . . . 577
READ . . . 578
READ . . . 578
REGISTER . . . 579
REGISTER . . . 579
RENAME . . . 580
RENAME . . . 580
RENUMBER . . . 581
RENUMBER . . . 581
RETURN . . . 582
RETURN . . . 582
ROUTINES . . . 583
ROUTINES . . . 583
RPCERR . . . 584
RPCERR . . . 584
RUN . . . 585
RUN . . . 585
program-name . . . 585
library-id . . . 585
SAVE . . . 586
SAVE . . . 586
object-name . . . 586
library-id . . . 586
SCAN . . . 587
SCAN . . . 587
SCAN for Mainframes . . . 587
General Rules . . . 587
Menu Options . . . 587
List Scan settings Found . . . 589
Direct Commands . . . 590
Batch Mode . . . 590
Subcommands . . . 591
Editing Rules . . . 591
SCAN under Natural Security . . . 592
SCAN for UNIX/OpenVMS . . . 593
SCAN Subcommands . . . 594
SCAN Editing Rules . . . 594
SCAN under Natural Security . . . 594
SCAN for Windows . . . 595
SCRATCH . . . 596
SCRATCH . . . 596
SETUP . . . 597
SETUP . . . 597
Syntax Explanation . . . 597
application-name . . . 597
command-name . . . 597
I Option . . . 597
SETUP/RETURN Example . . . 598
SQLERR . . . 599
SQLERR . . . 599
STOW . . . 600
STOW . . . 600
object-name . . . 600
library-id . . . 600

 Copyright © Software AG 2002

STRUCT 601
STRUCT 601
STRUCT for Mainframes 601
Generate Structured Source into Work Area 602
Partial Indentation 603
Display Structure of Source 604
Print Structure of Source 605
Write Structure of Source into Work Area 605
STRUCT for UNIX/OpenVMS 606
Partial Indentation 606
SYSADA - ADACALL 608
SYSADA - ADACALL 608
SYSBPM 609
SYSBPM 609
SYSDDM 610
SYSDDM 610
SYSEDT 611
SYSEDT 611
SYSERR 612
SYSERR 612
SYSEXT 613
SYSEXT 613
SYSEXT for Mainframes and UNIX/OpenVMS 613
SYSEXT for Windows 615
SYSFILE 616
SYSFILE 616
SYSFILE for Mainframes 616
SYSFILE for UNIX/OpenVMS and Windows 617
SYSMAIN 618
SYSMAIN 618
SYSNCP 619
SYSNCP 619
SYSOBJH 620
SYSOBJH 620
SYSPARM 621
SYSPARM 621
SYSPAUL 622
SYSPAUL 622
SYSPROD 623
SYSPROD 623
SYSPROD for Mainframes and UNIX/OpenVMS 623
SYSPROD for Windows 624
SYSPROF 625
SYSPROF 625
SYSPROF for Mainframes 625
SYSPROF for UNIX/OpenVMS 626
SYSPROF for Windows 627
SYSRPC 628
SYSRPC 628
SYSTP 629
SYSTP 629
SYSTRANS 630
SYSTRANS 630
SYSUNLD 631
SYSUNLD 631

TECH 632
TECH 632
TEST 633
TEST 633
UNCATALOG 634
UNCATALOG 634
UNLOCK 635
UNLOCK 635
UNMAP 636
UNMAP 636
UNREGISTER 637
UNREGISTER 637
UPDATE 638
UPDATE 638
XREF 639
XREF 639
Natural Security Considerations 640
Programming Reference 641
Programming Reference 641
General Information 642
General Information 642
User-Defined Variables 642
Naming Conventions 642
Definition of Variables 643
Statement Reference Notation - r 644
Definition of Format and Length 645
Special Formats 646
Index Notation 647
Referencing a Database Array 649
Referencing the Internal Count for a Database Array 650
Qualifying Data Structures 651
Constants 651
Numeric Constants 652
Alphanumeric Constants 652
Date and Time Constants 653
Hexadecimal Constants 654
Logical Constants 655
Floating Point Constants 655
Attribute Constants 655
Handle Constants 655
Report Specification - *rep* 656
Text Notation 656
User Comments 657
End of a Statement 657
Logical Condition Criteria 657
Relational Expression 658
Extended Relational Expression 660
MASK Option 661
SCAN Option 666
BREAK Within Logical Condition Criteria 667
IS Option - Checking Format and Length of Value 667
Evaluation of a Logical Variable 668
Modified Control Variables 668
SPECIFIED Option 669
Fields Used Within Logical Condition Criteria 669
Logical Operators in Complex Logical Expressions 670

 Copyright © Software AG 2002

Rules for Arithmetic Assignment . . . 671
Field Initialization . . . 671
Data Transfer . . . 671
Field Truncation and Field Rounding . . . 673
Result Format and Length in Arithmetic Operations . . . 673
Arithmetic Operations with Floating-Point Numbers . . . 674
Arithmetic Operations with Date and Time . . . 675
Performance Considerations for Mixed Format Expressions . . . 677
Precision of Results for Arithmetic Operations . . . 677
Error Conditions in Arithmetic Operations . . . 678
Processing of Arrays . . . 679
System Variables . . . 681
System Variables . . . 681
Alphabetical List of Variables . . . 681
*APPLIC-ID . . . 682
*APPLIC-NAME . . . 682
*COM . . . 682
*CONTROL . . . 683
*CONVID . . . 683
*COUNTER . . . 683
*CPU-TIME . . . 684
*CURS-COL . . . 684
*CURS-FIELD . . . 684
*CURS-LINE . . . 685
*CURSOR . . . 685
*DATA . . . 686
*DEVICE . . . 686
*DIALOG-ID . . . 687
*ERROR-LINE . . . 687
*ERROR-NR . . . 687
*ERROR-TA . . . 688
*ETID . . . 688
*EVENT . . . 688
*GROUP . . . 689
*HARDCOPY . . . 689
*HARDWARE . . . 689
*HOSTNAME . . . 689
*INIT-ID . . . 690
*INIT-PROGRAM . . . 690
*INIT-USER . . . 690
*ISN . . . 691
*LANGUAGE . . . 692
*LENGTH . . . 695
*LEVEL . . . 695
*LIBRARY-ID . . . 695
*LINE-COUNT . . . 695
*LINESIZE . . . 696
*LOG-LS . . . 696
*LOG-PS . . . 696
*MACHINE-CLASS . . . 696
*NATVERS . . . 697
*NET-USER . . . 697
*NUMBER . . . 697
*OCCURRENCE . . . 698
*OPSYS . . . 699
*OS . . . 699

*OSVERS 699
*PAGESIZE 700
*PAGE-NUMBER 700
*PARM-USER 701
*PATCH-LEVEL 701
*PF-KEY 701
*PF-NAME 702
*PID 702
*PROGRAM 702
*ROWCOUNT 702
*SCREEN-IO 703
*SERVER-TYPE 703
*STARTUP 704
*STEPLIB 705
*SUBROUTINE 705
*THIS-OBJECT 705
*TPSYS 706
*UI 706
*USER 707
*USER-NAME 707
*WINDOW-LS 707
*WINDOW-POS 707
*WINDOW-PS 707
*WINMGR 708
*WINMGRVERS 708
Date and Time System Variables 709
Large and Dynamic Variables/Fields 711
Large and Dynamic Variables/Fields 711
Introduction 711
Definition of Dynamic Variables 711
System Variable *LENGTH(field) 712
Size Limitation Checks 712
Statements EXPAND and REDUCE 712
Usage of Dynamic Variables 713
Assignments with Dynamic Variables 713
Initialization of Dynamic Variables 716
String Manipulation with Dynamic Alpha Variables 716
Logical Condition Criterion (LCC) with Dynamic Variables 717
Parameter Transfer with Dynamic Variables 719
Work File Access with Large and Dynamic Variables 723
DDM Generation and Editing for Varying Length Columns 723
Accessing Large Database Objects 724
Performance Aspects with Dynamic Variables 726
Natural System Functions 727
Natural System Functions 727
Using Natural System Functions 727
System Functions in SORT GIVE FUNCTIONS Statement 728
Arithmetic Overflows in AVER, NAVER, SUM or TOTAL 728
Statement Referencing *(r)* 728
Alphabetical List of Natural System Functions 729
AVER*(r)(field)* 729
COUNT*(r)(field)* 729
MAX*(r)(field)* 729
MIN*(r)(field)* 729
NAVER*(r)(field)* 729
NCOUNT*(r)(field)* 730

 Copyright © Software AG 2002

NMIN*(r)(field)* 730
OLD*(r)(field)* 730
SUM*(r)(field)* 730
TOTAL*(r)(field)* 730
Mathematical Functions 734
POS - Field Identification Function 737
RET - Return Code Function 738
SORTKEY - Sort-Key Function 739
Terminal Commands 741
Terminal Commands 741
General Considerations 741
Control Character 741
Entering a Terminal Command 741
Using Terminal Commands in Programs 741
Overview of Terminal Commands 742
Key Assignments 743
Terms Used 743
% - Continuation Indicator for INPUT in Batch 745
%A - Re-running a Recording 745
%B - Activate/De-activate the Recording Process 745
%B= - Recording Library 746
%C - Copying Contents of Page Buffer 746
%CS and %CC - Copying Data to Stack or *COM 747
%D - Activate Keyword/Delimiter Mode 748
%D= - Control of Outlining 748
%E - Display Screens Recorded with NATPAGE 749
%E= - Activate/De-activate Error Processing 750
%F - Activate Forms/Screen Mode 750
%F= - Frame Characters for Window 750
%G - Re-run Mode for Recording 751
%H - Hardcopy Output 751
%I - Record Current Screen with NATPAGE 753
%J - Invoke Helproutine 753
%K and %KP - Simulate PF- and PA-Key 753
%KN/%KO/%KS - Siemens Function-Key Logic 754
%L - Disable Lower- to Upper-Case Translation 754
%L= - Language Code 754
%M - Control of Message Line 755
%MSGSF - Display System Error Messages in Full 755
%N - Activate Non-Conversational Mode 756
%O - Terminate NATPAGE 756
%P - Activate NATPAGE for all Subsequent Screens 757
%P= - CALL Options 757
%P=S - Standard Linkage for Call 758
%P=V - Roll-Out for Call 758
%P=C - Pass Parameter Values instead of Address 758
%P=L - Call of LE/370 Dynamic Main Program 759
%Q - Suppress Map Printing in Batch Mode 759
%QO - Suppress Pseudo-Conversational Output 759
%QS - Simultaneous Output of Multiple Screens 760
%R - Repeat INPUT Statement 760
%R in a Recording - on Mainframe Computers only 760
%RM - Protection of Light Pen Sensitive Fields 761
%RN - Suppress Compression of Screen Data 761
%RO - Activate/De-activate Screen Optimization 762
%S - Resume NATPAGE 762

%T - Position Cursor to Top of Screen 763
%T*ll/cc* - Position Cursor to Line ll, Column cc 763
%T* - Position Cursor Outside Window 764
%T+/%T- - Position Cursor on Protected Fields 764
%TRE - Activate/Deactivate External Trace 765
%TRI - Activate/Deactivate Internal Trace 765
%T= - Activate Converter Routine for Device Type 766
%U - Translate Lower to Upper Case 766
%V - Control of Print Mode 767
%W - Window Processing 768
1. Window Size and Position on the Physical Screen 769
2. Window Position on a Logical Page 771
%WA and %WZ - Save Screen Image before Window 773
%X - Control of Statistics Line/Infoline 773
Infoline 774
Statistics Line - on Mainframe Computers only 774
%Y - Control of PF-Key Lines 776
Display Format of Function-Key Lines 776
Single- and Double-Line Display 776
Positioning of Function-Key Lines 777
Range of Displayed Function Keys 777
Intensified or Reverse Video Display of Function-Key Lines 777
Coloring of Function-Key Lines 777
Cursor-Sensitivity 777
%Z - Clear Source Area 779
%? - Help Information for Terminal Commands 779
%% and %. - Interrupt Current Operation 779
%% in Online Mode 779
%% in Batch Mode - on Mainframe Computers only 779
%. in Online and Batch Mode 780
%.P - Delete First Entry from Stack 781
%.S - Read Stack Entry Without Deleting it 781
%+ and %- - Natural Connection 782
%* - Inhibit Character Display 782
%* in Batch Mode - on Mainframe Computers only 782
%/ - End-of-File 783
%= - Assigning Colors to Fields 784
%<TECH - Display Technical Information 785
%<TEST - Invoke Debugging Facility 786
CLEAR Key - Interrupt Current Operation 786
CTRL+D Keys - Interrupt Current Operation 786
RESET+ENTER Keys - Attention Interrupt 786
Keywords and Reserved Words 787
Keywords and Reserved Words 787

 Copyright © Software AG 2002

Reference - Overview

This documentation applies to all platforms on which Natural can be used. It contains detailed information which you will find necessary when writing Natural applications.

- Parameter Reference
- Command Reference
- Programming Reference

Copyright © Software AG 2002

Parameter Reference

- Profile Parameters
- Session Parameters
- Parameter Modules (for Mainframes only)

General information on using parameters can be found in the platform-specific Operations documention.

 Copyright © Software AG 2002

Profile Parameters

Alphabetical List of Profile Parameters

Unlike the Natural session parameters whose use is consistent on all plattforms supported by Natural, many of the Natural profile parameters described in this document are not available on all platforms. Therefore, platform-specific details are marked accordingly in the parameter descriptions.

As the use of the Natural profile parameters is different on the various platforms, general information about parameter usage had to be included in the corresponding platform-specific documentation.

Natural for Mainframes

See the following documents in the Natural Operations for Mainframes documentation:

Profile Parameter Usage - Overview

- Natural Parameter Hierarchy
- Assignment of Parameter Values
- Profile Parameters Grouped by Function
- Using a Natural Parameter Module

Natural for Windows

See the following documents in the Natural Operations for Windows documentation:

Profile Parameter Usage

- Creating Parameter Files
- Profile Parameters Grouped by Function
- Dynamic Assignment of Parameter Values
- Runtime Assignment of Parameter Values

Natural for UNIX/OpenVMS

See the following documents in the Natural Operations for UNIX/OpenVMS documentation:

Profile Parameters

- General Information
- Creating Parameter Files
- Profile Parameters - Overview
- Dynamic Assignment of Parameter Values
- Runtime Assignment of Parameter Values

Copyright © Software AG 2002

ACIPATT - Define Node Pattern for ACI Protocol

This Natural profile parameter is for UNIX/OpenVMS and Windows platforms only.

Note for mainframes:
Use the keyword parameter ACIPATT of the NTRPC macro for static specification, or the subparameter with the same name of the profile parameter RPC.

This parameter determines which node names will be used for the ACI transport protocol together with Natural Remote Procedure Call. You can specify a pattern to match node names used for the ACI transport protocol (that is EntireX Broker nodes).

ACIPATT is specified on the client side only. It applies only to nodes which were defined in the server maintenance of the SYSRPC utility (because there the transport protocol cannot be specified).

Possible settings	1-8 characters, including *	Pattern for node name matching. All node names matching this pattern are assumed to be ACI nodes.
Default setting	*	All node names match the pattern. The ACI transport protocol is used for all of them.
Dynamic specification	YES	
Specification within session	YES	At runtime, this value can be overwritten by the parameter maintenance of the SYSRPC utility.

Node names beginning with the letter(s) specified and/or up to the first "*" are matched. For example, if you specify N*, all node names beginning with N are regcognized as ACI nodes.

 Copyright © Software AG 2002

 ACIVERS - Define ACI Version for Use with EntireX Broker ACI

ACIVERS - Define ACI Version for Use with EntireX Broker ACI

This Natural profile parameter is for UNIX/OpenVMS and Windows platforms only.

Note for mainframes:
Use the keyword parameter ACIVERS of the NTRPC macro for static specification, or the subparameter with the same name of the profile parameter RPC.

It specifies the API version to be used with the EntireX Broker ACI. The broker stub in use must support the ACI version defined here. The setting of ACIVERS enables special features of the EntireX Broker depending on the API version you are using. For details see Setting Up an EntireX Broker Access in the Natural Remote Procedure Call documentation.

ACIVERS can be specified on both the client and the server side.

Possible settings	1 - 5	Single-digit number. The higher the version, the more features are available.
Default setting	2	API version 2 is used.
Dynamic specification	YES	
Specification within session	NO	

For further information see the Natural Remote Procedure Call documentation.

Copyright © Software AG 2002

 ACTPOLICY - Defines Default Activation Policy for DCOM Classes

ACTPOLICY - Defines Default Activation Policy for DCOM Classes

This Natural profile parameter is for UNIX/OpenVMS and Windows platforms only. For mainframes, see the subparameter ACTPOL of the profile parameter DCOM.

It defines the default activation policy for DCOM classes. It is evaluated if a Natural class is registered as a DCOM class with the REGISTER command.

Possible settings	ES	External single activation is performed.
	EM	External multiple activation is performed.
	IM	Internal multiple activation is performed.
Default setting	EM	
Dynamic specification	YES	
Specification within session	NO	

 Copyright © Software AG 2002

ADAMODE - Adabas Interface Mode

This Natural profile parameter is for mainframes only.

It controls the number of Adabas user queue elements (UQE) per Natural session.

Possible settings	0	Start Natural with one Adabas session without using the Adabas X48 communication (Natural in Natural Version 2.2 mode). Only one UQE is initialized and the ID of the UQE is built by the ADALN*x* module. All database calls - either sent by the nucleus, an application program or a 3-GL program - are considered as the same Adabas user. Running under SYSPLEX is not possible.
	1	Start Natural with one user and Adabas X48 communication. Only one UQE is initialized, all nucleus and application database calls are submitted for the same UQE, however, calls sent by 3-GL programs are excluded. Running under SYSPLEX is possible.
	2	Start Natural with two Adabas sessions, both using the Adabas X48 communication (Natural Version 2.3 mode). Two UQEs are generated at Natural session startup, and nucleus and application calls are running separate from each other. Database calls sent by 3-GL programs are excluded from Natural transactions. Running under SYSPLEX is possible.
	3	Start Natural with one UQE using Adabas X48 communication for nucleus calls and one UQE without Adabas X48 communication for application calls. Nucleus and application calls are running separate from each other. Database calls sent by 3-GL programs are included into Natural application transactions. Running under SYSPLEX is not possible.
Default setting	2	
Dynamic specification	YES	
Specification within session	NO	

Background Information:

With **Natural Version 2.2**, every Natural session is related to one Adabas user queue element (UQE). All calls to Adabas - sent by either the nucleus (e.g. to load Natural objects for execution) or any user application - appear to Adabas as the same user. The user identification in the UQE is automatically built by the ADALNx module and - depending on the environment (CICS, BS2000/OSD, TSO, etc.) - contains the task number, the terminal ID or any other unique item. Although the uniqueness of nucleus/application works fine in most cases, problems may arise under the following conditions:

When the nucleus and an application are starting update transactions (UDATE, STORE, DELETE) at the same time, an ET/BT command sent either by the nucleus or the application will confirm or roll back all transactions, no matter to whom they originally belong.

When a timeout situation (NAT3009) occurs, Natural has to inform the session user, but the logical user (nucleus/application) for whom the timeout was sent is not known.

With **Natural Version 2.3 and above**, a new logic to communicate (X48) with Adabas was introduced. This enables you to set up different users (UQEs) on Adabas for one Natural session. This feature separates Natural nucleus calls from application calls, and all transactions of the nucleus and the applications are running in their own logical environment. A timeout situation (NAT3009) received for nucleus calls is automatically handled by Natural and does not lead to an error message displayed to the user.

To allow Natural to run in a Sysplex environment, the mechanism previously used to build a UQE cannot be used any longer, because task number, terminal ID or whatever used by ADALNx to build a UQE ceases to be unique, since these items may change in a SYSPLEX environment on every terminal I/O statement. The X48 communication with Adabas is a prerequisite for Natural to run in a SYSPLEX environment.

It has the minor drawback every Natural session is related to two different UQEs on Adabas; this means, the number of Adabas UQEs is doubled compared with Natural Version 2.2. This may cause problems if the number of UQEs is limited (refer to the Adabas parameter NU).

 Copyright © Software AG 2002

ADANAME - Name of Adabas Link Routine

This Natural profile parameter is for mainframes only, but does not apply for UTM and Com-plete.

It specifies the name of the Adabas link routine to be used.

Possible settings	1 - 8 characters	Valid module or entry name.
Default setting	ADABAS	
Dynamic specification	YES	
Specification within session	NO	

If the Adabas link routine is linked to the Natural parameter module (NATPARM) and its entry name is the same as the one specified by ADANAME in the parameter module, the linked routine will be used. If not, the specified link routine will be loaded dynamically. Thus, it is no longer necessary to statically link the Adabas link module to the Natural nucleus.

It is possible to run the same Natural nucleus with different Adabas link modules.

Copyright © Software AG 2002

ADAPRM - REVIEW/DB Support

This Natural profile parameter is for mainframes only.

It is used to pass Natural-session data to REVIEW/DB within the seventh Adabas buffer.

Possible settings	ON	Natural session data is passed. Set ADAPRM to ON if REVIEW/DB is installed.
	OFF	No Natural session data is passed.
Default setting	OFF	
Dynamic specification	YES	
Specification within session	NO	

 Copyright © Software AG 2002

ADASBV - Adabas Security By Setting

This Natural profile parameter is for mainframes only.

It can be used to prevent invalid results for accesses to Adabas files that are protected by "security-by-setting". When a file that is protected by "security-by-setting" is accessed, invalid results may be returned in some cases where no format buffer is generated and passed to Adabas.

Possible settings	ON	Natural session data is passed. It is recommended that you set ADASBV=ON if you access "security-by-setting"-protected Adabas files. A format buffer is then always passed to Adabas for a database access (even if this be a 2-byte dummy buffer), thus avoiding invalid results.
	OFF	No Natural session data is passed.
Default setting	OFF	
Dynamic specification	YES	
Specification within session	NO	

ASIZE - Entire System Server Auxiliary Buffer

This Natural profile parameter is for mainframes only.

It only applies if Entire System Server is installed.

It determines the size of the Entire System Server auxiliary buffer.

Possible settings	0 - 64	Buffer size in KB. If Entire System Server is to be used, this parameter **must** be set; see the Entire System Server documentation.
Default setting	0	If ASIZE=0 is specified or if the requested space is not available, the Entire System Server is not activated.
Dynamic specification	YES	
Specification within session	NO	

 Copyright © Software AG 2002

ASPSIZE (Internal Use)

This Natural profile parameter is for mainframes only.

It is reserved for internal use by Natural. Do not change its setting.

ASYNNAM - Output System ID for Asynchronous Processing

This Natural profile parameter is for mainframes only. It only applies under UTM.

For asynchronous processing between two Natural/UTM applications, this parameter specifies the address of the synchronous application which is used by the asynchronous application to send messages to the synchronous application.

Possible settings	1 - 8 characters	Valid transaction name.
Default setting	blank	No asynchronous processing takes place.
Dynamic specification	YES	
Specification within session	NO	

For further information on asynchronous processing under UTM, see **Asynchronous Transaction Processing** in the Natural TP Monitor Interfaces documentation.

 Copyright © Software AG 2002

ATTN - Attention Key Interrupt Support

This Natural profile parameter is for mainframes only.

It controls the use of the attention key for IBM SNA terminals. Pressing the attention key can interrupt Natural processing with an appropriate error message (NAT1016). The availability of an attention key depends on the environment and on the terminal type.

This functionality is also available for Natural in Batch under VSE/ESA.

Possible settings	ON	The attention key causes Natural processing to be interrupted.
	OFF	The attention key is ignored.
Default setting	ON	
Dynamic specification	YES	
Specification within session	NO	

AUTO - Automatic Logon

This Natural profile parameter is for:

- mainframes
- UNIX/OpenVMS and Windows

It causes an automatic logon to a specific library at the start of the Natural session.

Possible settings	ON	An automatic logon is executed at the start of the Natural session.
	OFF	No automatic logon is performed.
Default setting	OFF	
Dynamic specification	YES	
Specification within session	NO	

Mainframes

The setting contained in the system variable *INIT-USER is used as the user ID for the logon.

Note:
If used with Natural Security, AUTO=ON disables logons with another user ID (see your Natural Security documentation for further information).

UNIX/OpenVMS and Windows

The setting of the profile parameter INIT-LIB is used as library ID for the logon.

Note:
If used with Natural Security, AUTO=ON disables logons with another user ID, and the INIT-LIB parameter is not evaluated (see the Natural Security documentation for further information).

 Copyright © Software AG 2002

AUTOREGISTER - Automatic Registry Update

This Natural profile parameter is for UNIX/OpenVMS and Windows platforms only. For mainframes see the subparameter AUTOREG of the profile parameter DCOM.

It determines whether Natural classes are to be registered as DCOM classes each time they are cataloged. Alternatively, Natural classes can be registered manually with the REGISTER command. This parameter is used by NaturalX.

Possible settings	ON	Registry will be updated automatically.
	OFF	Registry will not be updated automatically.
Default setting	OFF	
Dynamic specification	YES	
Specification within session	NO	

AUTORPC - Automatic Natural RPC Execution

This Natural profile parameter is for UNIX/OpenVMS and Windows platforms only.

Note for mainframes:
Use the keyword parameter AUTORPC of the NTRPC macro for static specification, or the subparameter with the same name of the profile parameter RPC.

This parameter determines whether or not Natural RPC will automatically try to execute a subprogram remotely (on the server side) which was not found locally (on the client side). For details see Stubs and Automatic RPC in the Natural Remote Procedure Call documentation.

AUTORPC is specified on the client side only.

Possible settings	ON	Natural RPC will automatically try to execute it remotely.
	OFF	Natural RPC will not automatically try to execute it remotely. If AUTORPC=OFF, you can execute CALLNATs remotely using stubs.
Default setting	OFF	
Dynamic specification	YES	
Specification within session	NO	

For further information see the Natural Remote Procedure Call documentation.

Note:
The former parameter AUTOREMOTE is no longer supported.

 Copyright © Software AG 2002

BATCH - Batch Mode Simulation

This Natural profile parameter is for UNIX/OpenVMS and Windows platforms only.

It sets the system variable *DEVICE to "BATCH" when Natural is started.

Possible settings	specified	When started with parameter BATCH, error messages under Windows 98/ME/NT/2000 are not displayed, but written to a log file. The log file is named natbatch.log and is located in the Natural binary directory.
Default setting	not specified	Error messages under Windows 98/ME/NT/2000 are displayed but not written to a log file.
Dynamic specification	YES	This parameter can only be specified dynamically.
Specification within session	NO	

To run Natural in real batch mode use parameter BATCHMODE instead of BATCH.

Copyright © Software AG 2002

BATCHMODE - Real Batch Mode

This Natural profile parameter is for UNIX/OpenVMS and Windows platforms and applies to batch mode only.

It enables real batch mode and sets the system variable *DEVICE to "BATCH" when Natural is started.

Possible settings	specified	Natural will run in batch mode.
Default setting	not specified	Natural will run in interactive mode, or in batch mode simulation.
Dynamic specification	YES	The parameter can only be specified dynamically.
Specification within session	NO	

 Copyright © Software AG 2002

BMBLANK - Display Trailing Blanks

This Natural profile parameter is for UNIX/OpenVMS and Windows platforms and applies to batch mode only.

It is used to control the display of trailing blanks in the batch output file CMPRINT.

This parameter applies only if the parameter BMSIM is set to MF. Trailing blanks are generated automatically if BMSIM is set to MF. BMBLANK has no effect if BMSIM is set to OS or VM.

Possible settings	ON	Trailing blanks are written to CMPRINT.
	OFF	No trailing blanks are written to CMPRINT.
Default setting	ON	
Dynamic specification	YES	
Specification within session	NO	

BMCONTROL - Display Control Characters

This Natural profile parameter is for UNIX/OpenVMS and Windows platforms and applies to batch mode only.

It controls the output of control characters (such as form feed and line feed) in the batch output file CMPRINT.

Possible settings	ON	Control characters will be written to CMPRINT.
	OFF	No control characters will be written to CMPRINT.
Default setting	ON	
Dynamic specification	YES	
Specification within session	NO	

 Copyright © Software AG 2002

BMFRAME - Window Frame Characters

This Natural profile parameter is for UNIX/OpenVMS and Windows platforms and applies to batch mode only.

With this parameter you can define window frame characters that will be written to the batch output file CMPRINT. Specify a sequence of 6 characters.

Possible settings	6 characters	Character Position:	Displays:
		1 2 3 4 5 6	Horizontal bar Vertical bar Upper-left corner Upper-right corner Lower-left corner Lower-right corner
Default setting	"-!++++"		
Dynamic specification	YES		
Specification within session	NO		

Example:

To define the following frame you have to specify BFRAME=123456.

```
3111111111114
2            2
2            2
5111111111116
```

BMSIM - Similar Batch Mode Output

This Natural profile parameter is for UNIX/OpenVMS and Windows platforms and applies to batch mode only.

It is used for the general appearance description of the batch mode output file CMPRINT.

Possible settings	MF	Forces output similar to Natural for Mainframe Version 2.3. Each line in CMPRINT is filled with trailing blanks. A control character appears at the beginning of each line of CMPRINT. The control character codes are similar to the IBM control character option ASA. The following control character codes are used: **Control Code** — **Interpretation** blank — Normal output line without control characters 0 — Insert one empty line - — Insert two empty lines + — Print this line twice (bold printing)
	VM	Forces output similar to Natural for OpenVMS Version 2.1. The following control character codes are used: **Control Code** — **Interpretation** \n — Line feed before printing this line \f — Form feed before printing this line The FIN command will be generated if it is not specified in the batch command file CMSYNIN.
	OS	Forces output similar to Natural for UNIX or OpenVMS. The following control character codes are used: **Control Code** — **Interpretation** \n — Line feed before printing this line \f — Form feed before printing this line
Default setting	OS	
Dynamic specification	YES	
Specification within session	NO	

 Copyright © Software AG 2002

BMTIME - Display Process Time

This Natural profile parameter is for UNIX/OpenVMS and Windows platforms and applies to batch mode only.

It is used to display the elapsed and used CPU time consumed by the Natural process. This output will be written to the end of the batch output file CMPRINT.

Possible settings	ON	The elapsed and used CPU time is written to the end of the batch output file.
	OFF	The output is not written to the batch output file.
Default setting	OFF	
Dynamic specification	YES	
Specification within session	NO	

The time format is as follows:

DDDxHH:*II*:*SS*.*UU*

where:

- *DDD* is the number of days (at maximum 999)
- *x* is " " (blank) if *DDD* is less or equal to 999,
  or "+" (plus sign) if *DDD* is greater than 999
- *HH* is the number of hours
- *II* is the number of minutes
- *SS* is the number of seconds
- *UU* is the number of hundredths of seconds

Example:

```
Used CPU time:   0 00:00:00.56
Elapsed time:    0 00:00:16.20
```

BMTITLE - Display Window Title

This Natural profile parameter is for UNIX/OpenVMS and Windows platforms and applies to batch mode only.

It is used to control the displaying of window titles in the batch output file CMPRINT.

Possible settings	ON	A window title will be displayed in CMPRINT.
	OFF	No window title will be displayed in CMPRINT.
Default setting	ON	
Dynamic specification	YES	
Specification within session	NO	

 Copyright © Software AG 2002

BMVERSION - Display Natural Version

This Natural profile parameter is for UNIX/OpenVMS and Windows platforms only.

It is used to control the display of the Natural version including the startup and termination time.

Possible settings	ON	The Natural version and startup time are written to the very first line of the batch output file CMPRINT, the termination time is written at the end of CMPRINT.
	OFF	The Natural version and startup time are not written to CMPRINT.
Default setting	OFF	
Dynamic specification	YES	
Specification within session	NO	

Example:

```
Natural V 5.1.1  Software AG 2001 started on 23-Jul-2001 at 09:03:36.34
 ...
Natural terminated on 23-Jul-2001 at 09:03:36.38
```

BPI - Buffer Pool Initialization

This Natural profile parameter is for:

- mainframes
- UNIX/OpenVMS

Possible settings	See NTBPI Macro.	
Default setting	none	
Dynamic specification	YES	This parameter can only be specified dynamically.
Specification within session	NO	

It corresponds to the NTBPI macro in the parameter module and can be used to specify dynamically the same options which you can specify statically with the NTBPI macro.

The BPI parameter is specified as follows:

BPI=(TYPE=*type*,SEQ=*n*,NAME=*name*,SIZE=*nnn*,LIST=*name*,TXTSIZE=*n*)

- To dynamically deactivate a buffer-pool definition, you use the keyword OFF:
  BPI=(TYPE=*type*,SEQ=*n*,OFF)
- If OFF is used, it must be specified after TYPE and SEQ.
- If you use the BPI parameter to update an existing buffer pool definition in the parameter module, you must specify new settings in all those subparameters which are to be changed; if you do not, the old settings will be used.
  If, for example, you want to change from a global to a local buffer pool, you must specify NAME=' '.
- If you use the BPI parameter to dynamically specify a new backup buffer pool, you must specify a sequence number (SEQ) for it.
  If you omit the SEQ specification, the definition of the primary buffer pool (SEQ=0) will be overwritten.
- The NAME, SIZE, LIST and TXTSIZE specifications for the primary buffer pool (SEQ=0) can also be set dynamically with the profile parameters BPNAME, BPSIZE, BPLIST and BPTEXT.
- For general information on the Natural buffer pool, see Natural Buffer Pool.
- For the Software AG Editor buffer pool, see also the profile parameter EDPSIZE.

Examples:

```
/* TWO (ALTERNATIVE) BUFFER POOL INITIALIZATIONS
NAT23 BPI=(SEQ=0,NAME=LBP1),BPI=(SEQ=1,NAME=LBP2),BPI=(SEQ=2,SIZE=500)
NAT23 BPI=(SEQ=0,TYPE=EDITOR,NAME=LBPE1),BPI=(SEQ=1,TYPE=EDITOR,SIZE=500)
```

In the first example, Natural first tries to allocate a global Natural buffer pool with the name "LBP1"; if this buffer pool is not found, it tries to allocate "LBP2"; if this is not found, it allocates a local buffer pool with a size of 500 KB.

In the second example, Natural first tries to allocate a global editor buffer pool with the name "LBPE1"; if this is not found, it allocates a local editor buffer pool with a size of 500 KB.

 Copyright © Software AG 2002

BPID - Specify Buffer Pool ID

This Natural profile parameter is for UNIX/OpenVMS and Windows platforms only.

It specifies the name (ID) of the Natural buffer pool.

Possible settings	1 - 8 characters	Name of the Natural buffer pool.
Default setting	NATBP	
Dynamic specification	YES	
Specification within session	NO	

BPLIST - Name of Preload List for Natural Buffer Pool

This Natural profile parameter is for:

- mainframes
- UNIX/OpenVMS

With BPLIST you can specify the name of a preload list to be used for the Natural buffer pool.

Possible settings	1-8 characters, or blank	Name of a preload list to be used for the Natural buffer pool.
Default setting	blank	No preload list is used.
Dynamic specification	YES	On mainframes this parameter can only be specified dynamically.
	NO	No dynamic specification on UNIX/OpenVMS.
Specification within session	NO	

For general information, see Natural Buffer Pool.

Mainframe-specific Information

The parameter corresponds to the LIST specification of the BPI profile parameter or the NTBPI macro.

It only applies to the primary Natural buffer pool (TYPE=NAT, SEQ=0). If there is a primary buffer pool with SEQ=0 in NATPARM, only the LIST setting of this buffer pool is updated. Internally, the BPLIST specification is converted into an equivalent BPI specification.

Example:
`BPLIST=LIST3` is converted into: `BPI=(TYPE=NAT,SEQ=0,LIST=LIST3)`

 Copyright © Software AG 2002

BPNAME - Name of Natural Global Buffer Pool

This Natural profile parameter is for:

- mainframes
- UNIX/OpenVMS

With this parameter, you can specify the name of the Natural global buffer pool.

Possible settings	1 - 8 characters, or blank	Name of the Natural global buffer pool.
Default setting	blank	To use a *local* Natural buffer pool.
Dynamic specification	YES	This parameter can only be specified dynamically.
Specification within session	NO	

For general information, see Natural Global Buffer Pool.

Mainframe-specific Information

This parameter can only be specified dynamically. It corresponds to the NAME specification of the BPI profile parameter or the NTBPI Macro respectively.

The BPNAME parameter only applies to the primary Natural global buffer pool (TYPE=NAT, SEQ=0).If there is a primary buffer pool with SEQ=0 in NATPARM, only the NAME setting of this buffer pool is updated. Internally, the BPNAME specification is converted into an equivalent BPI specification.

Example:
`BPNAME=GBP1` is converted into: `BPI=(TYPE=NAT,SEQ=0,NAME=GBP1)`

BPNLE - Number of Directory Entries

This Natural profile parameter is for UNIX/OpenVMS and Windows platforms only.

It determines the number of directory entries. This value is used by the NATBPSRV server during the start of the buffer pool. If BPNLE has reached the maximum value, this value is automatically increased.

Possible settings	10 - 9999	Number of directory entries
Default setting	10	
Dynamic specification		
Specification within session		

 Copyright © Software AG 2002

BPPROP - Global Buffer Pool Propagation

This Natural profile parameter is for:

- mainframes (only applies under OS/390 and BS2000/OSD)
- UNIX/OpenVMS

It controls the propagation of changes to an object in a buffer pool. If a modification occurs affecting a Natural object residing in one (global or local) buffer pool, this modification can be propagated to other global buffer pools - this will ensure the consistency of the buffer pools.

Possible settings	OFF	Changes are not propagated to any other global buffer pool. **Note for OS/390:** Any setting other than OFF requires that the Authorized-Services Manager is active.
	GLOBAL	Changes are propagated to all other global buffer pools. In an OS/390 Sysplex environment, the changes are only propagated within the current OS/390 image. (*)
	PLEX	Changes are propagated to all other global buffer pools of the same name within the entire OS/390 Sysplex. (*)
	GPLEX	Changes are propagated to all other global buffer pools within the entire OS/390 Sysplex. (*) **Note for BS2000/OSD:** The setting GPLEX has the same effect as GLOBAL.
Default setting	OFF	
Dynamic specification	YES	
Specification within session	NO	

* Under OS/390, the propagation is always restricted to the Natural subsystem in which the change has occurred; that is, the scope of the propagation, as set with the BPPROP parameter, applies only within that subsystem, but not to other subsystems. For details, see Natural Subsystem.

For further information on the propagation, see Natural Global Buffer Pool.

BPSFI - Object Search First in Buffer Pool

This Natural profile parameter is for:

- mainframes
- UNIX/OpenVMS and Windows

With this parameter, you determine the sequence in which a requested object that is to be executed is searched for in the buffer pool and in the system file(s). You can choose between two search sequences:

Possible settings	ON	Search Sequence 1 is used (search buffer pool first for all libraries, then the system file(s)). Natural looks for the object in the following sequence until it is found: 1. in the buffer pool, first in the current library, then in one steplib after another, then in the two SYSTEM libraries; 2. in the system file(s), first in the current library, then in one steplib after another, then in the two SYSTEM libraries.
	OFF	Search Sequence 2 is used (alternating search in buffer pool and system file(s) for each library). Natural looks for the object in the following sequence until it is found: 1. in the current library, first in the buffer pool, then in the system file(s); 2. in one steplib after another, first in the buffer pool, then in the system file(s) for each steplib; 3. in the two SYSTEM libraries, first in the buffer pool, then in the system file(s) for each library.
Default setting	OFF	
Dynamic specification	YES	
Specification within session	NO	

 Copyright © Software AG 2002

BPSIZE - Size of Natural Local Buffer Pool

This Natural profile parameter is for:

- mainframes (except Com-plete and BS2000/OSD, see below)
- UNIX/OpenVMS and Windows

Mainframes

It specifies the size of the Natural local buffer pool. It corresponds to the SIZE specification of the BPI profile parameter or the NTBPI macro.

Possible settings	100 - 2097151	Size of the Natural local buffer pool in KB.
Default setting	256	
Dynamic specification	YES	This parameter can only be specified dynamically.
Specification within session	NO	

BPSIZE only applies to the primary Natural local buffer pool (TYPE=NAT, SEQ=0). For a global buffer pool, it is ignored.If there is a primary buffer pool with SEQ=0 in NATPARM, only the SIZE setting of this buffer pool is updated.

In multi-user environments (for example, under CICS), the BPSIZE parameter only affects the very first Natural session, which initializes the local buffer pool.

Internally, the BPSIZE specification is converted into an equivalent BPI specification.

Example:
`BPSIZE=1500` is converted into: `BPI=(TYPE=NAT,SEQ=0,SIZE=1500)`

For general information, see Natural Buffer Pool.

Platform:	**Comment:**
Com-plete	The size of a local buffer pool is set as described in the Natural Installation Guide for Mainframes.
BS2000/OSD	The size of a local buffer pool is specified with the parameter SIZE.

UNIX/OpenVMS and Windows

This setting is used during startup of the Natural buffer pool by the NATBPSRV server to create a global shared memory segment of the specified size.

Possible settings	1 - 512	Size of the Natural local buffer pool in MB.
Default setting	1	
Dynamic specification	NO	
Specification within session	NO	

BPTEXT - Size of Text Segments in Natural Buffer Pool

This Natural profile parameter is for mainframes only.

It specifies the size of the segments into which the text pool area of the Natural buffer pool is divided. It corresponds to the TXTSIZE specification of the BPI profile parameter or the NTBPI macro respectively.

Possible settings	1, 2, 4	Size of segments in KB.
Default setting	4	
Dynamic specification	YES	This parameter can only be specified dynamically.
Specification within session	NO	

The BPTEXT parameter only applies to the primary Natural buffer pool (TYPE=NAT, SEQ=0). In the case of a global buffer pool, it is ignored. If there is a primary buffer pool with SEQ=0 in NATPARM, only the TXTSIZE setting of this buffer pool is updated.

In multi-user environments (for example, under CICS), the BPTEXT parameter only affects the very first Natural session, which initializes the local buffer pool.

Internally, the BPTEXT specification is converted into an equivalent BPI specification.

Example:
`BPTEXT=4` is converted into: `BPI=(TYPE=NAT,SEQ=0,TXTSIZE=4)`

For general information on the Natural Buffer pool, see Natural Buffer Pool.

 Copyright © Software AG 2002

BSIZE - Size of EntireX Broker Buffer

This Natural profile parameter is for mainframes only.

It only applies if EntireX Broker is installed. Currently, if EntireX Broker is used, EntireX Broker specifies the buffer size automatically.

Possible settings	0 - 64	Buffer size in KB.
Default setting	0	
Dynamic specification	YES	
Specification within session	NO	

CANCEL - Session Cancellation with Dump

This Natural profile parameter is for mainframes only.

With this parameter, you can specify a character string that will cause the Natural session to be terminated with a dump. This may be useful for debugging purposes.

Possible settings	1 to 8 characters	When you enter this character string in any input field within your Natural session (beginning in the first input field), the session will be terminated immediately and a dump will be produced.
Default setting	*CANCEL	
Dynamic specification	YES	
Specification within session	NO	

 Copyright © Software AG 2002

CC - Error Processing in Batch Mode

This Natural profile parameter is for:

- mainframes
- UNIX/OpenVMS and Windows

It determines the action to be taken if an error is detected during the compilation/execution of a Natural program in batch mode. It only applies in batch mode.

It does not apply if user-written error-handling routines are used.

Mainframes

Possible settings	ON	Natural flushes the input data stream for SYNIN and OBJIN until a line containing "%%" in the first two positions is encountered or until an end-of-file condition is detected. If more data are available in the input stream, Natural resumes reading after the line containing "%%".
	OFF	Natural attempts to process the next program (or command) in the input stream.
Default setting	OFF	
Dynamic specification	YES	
Specification within session	YES	Within a Natural session, the profile parameter CC can be overridden by the **session parameter** CC.

When a Natural session terminates, Return Code 4 is passed to the invoking program with register 15 if an error is detected (regardless of the CC setting).

UNIX/OpenVMS and Windows

Possible settings	ON	Natural flushes the input data stream for CMSYNIN and CMOBJIN until a line containing "%%" in the first two positions is encountered or until an end-of-file condition is detected. If more data are available in the input stream, Natural resumes reading after the line containing "%%".
	OFF	Natural attempts to process the next program (or command) in the input stream. If all input in processed, Natural terminates with return code 61 and writes the Natural error 9987 (Error occurred during execution/compilation.) to the batch output file (if ENDMSG is ON).
Default setting	OFF	
Dynamic specification	YES	
Specification within session	YES	Within a Natural session, the profile parameter CC can be overridden by the **session parameter** CC.

CCTAB - Printer Escape-Sequence Control Character

This Natural profile parameter is for mainframes only.

It corresponds to the NTCCTAB macro in the parameter module. It can be used to specify dynamically the same options which you can specify statically with the NTCCTAB Macro.

Possible settings	see NTCCTAB Macro	
Default setting	see NTCCTAB Macro	
Dynamic specification	YES	
Specification within session	NO	

Dynamic specification works as follows:

`CCTAB=(`*`name`*`,`*`subparameters`*`)`

For each profile, a separate CCTAB must be specified.

If you specify subparameters, the following rules apply:

- OPN and CLS can be specified in any sequence.
- The CS subparameter must follow the CODE subparameter.
- The CSS and CSE subparameters must follow the ATR subparameter.

Example:

```
CCTAB=(DBCST,CODE=OE,CS=400E,CODE=OF,CS=0F40,ATR=P5DBCS,CSS=OE,CSE=OF),
CCTAB=(TEST,OPN='FIRST LINE MESSAGE',CLS='LAST LINE MESSAGE')
```

 Copyright © Software AG 2002

CDYNAM - Dynamic Loading of Non-Natural Programs

This Natural profile parameter is for:

- mainframes
- UNIX/OpenVMS and Windows

Mainframes

This parameter determines how many non-Natural programs can be loaded dynamically by Natural during the execution of a single Natural program.

Possible settings	0 - 1024	Determines the maximum number of non-Natural programs which can be loaded per Natural program. If CDYNAM=0, no dynamic loading of non-Natural programs will be performed by Natural.
Default setting	5	
Dynamic specification	YES	
Specification within session	NO	

UNIX/OpenVMS and Windows

This parameter indicates whether or not non-Natural programs are to be loaded dynamically by Natural.

Possible settings	ON	Any number of non-Natural programs can be loaded dynamically during the execution of a Natural program.
	OFF	Dynamic loading of non-Natural programs is not performed by Natural.
Default setting	ON	
Dynamic specification	YES	
Specification within session	NO	

Copyright © Software AG 2002

CF - Character for Terminal Commands

This Natural profile parameter is for:

- mainframes
- UNIX/OpenVMS and Windows

It specifies the control character for Natural terminal commands; that is, the character which is to be used as the first character of any terminal command.

Possible settings	any special character	A terminal command must begin with the character specified.
	OFF	No control character for terminal commands is available. Terminal commands issued with SET CONTROL statements, however, are still accepted.
Default setting	%	A terminal command must begin with the character "%".
Dynamic specification	YES	
Specification within session	YES	Within a Natural session, the profile parameter CF can be overridden by the **session parameter** CF .

Mainframes

The character specified with the CF parameter must not be the same as the one specified with the HI parameter (help character) or IA parameter (input assign character); it should not be the same as the one specified with the DC parameter (decimal character) or ID parameter (input delimiter character).

UNIX/OpenVMS and Windows

The character specified with the CF parameter must not be the same as the one specified with the IA parameter (input assign character), the DC parameter (decimal character) or the ID parameter (input delimiter character); it should not be the same as the one specified with the HI parameter (help character).

 Copyright © Software AG 2002

CFWSIZE (Internal Use)

This Natural profile parameter is for mainframes only.

This parameter is reserved for internal use by Natural. Do not change its setting.

CLEAR - Processing of CLEAR Key in NEXT Mode

This Natural profile parameter is for:

- mainframes
- UNIX/OpenVMS and Windows

It causes Natural to execute a specific Terminal Command whenever CLEAR is pressed during program execution in NEXT mode.

Possible settings	any character	The default action can be overridden by supplying a character which, when appended to the terminal-command control character (as specified with the CF parameter), forms a valid Natural terminal command.
		
		Example:
		`CF=%`
		`CLEAR=R`
		Natural executes the terminal command "%R" when CLEAR is pressed in NEXT mode.

Default setting	%	By default, when the CLEAR key is pressed, Natural responds as if the user had entered the terminal command "%%".
Dynamic specification	YES	
Specification within session	NO	

 Copyright © Software AG 2002

CM - Command Mode

This Natural profile parameter is for:

- mainframes
- UNIX/OpenVMS and Windows

It can be used to suppress Natural command mode (NEXT and MORE).

Possible settings	ON	NEXT and MORE are available for command input.
	OFF	The Natural session will be terminated whenever NEXT is encountered; the MORE line will be write-protected (no input possible).
Default setting	ON	
Dynamic specification	NO	For UNIX/OpenVMS and Windows.
	YES	For Mainframes.
Specification within session	NO	

 CMOBJIN - Batch Input File for Natural INPUT Data

CMOBJIN - Batch Input File for Natural INPUT Data

This Natural profile parameter is for UNIX/OpenVMS and Windows. It applies to batch mode only.

It is used for data intended to be read by Natural INPUT statements. These types of data can alternatively be placed in the CMSYNIN file immediately following the relevant RUN or EXECUTE command. The number of characters actually processed is restricted to 512 characters per line.

Possible settings	any string	**Example for OpenVMS:** CMOBJIN = ALF$USER:[home.tmp]batch.inp **Example for UNIX:** CMOBJIN = $HOME/tmp/batch.inp **Example for Windows:** CMOBJIN=C:\tmp\data.txt
Default setting	none	
Dynamic specification	YES	
Specification within session	NO	

If the setting for the profile parameter CMSYNIN is equal to the setting of CMOBJIN, Natural reads input from CMSYNIN.

If an error occurs, Natural reacts in accordance with the setting of the parameter CC.

 Copyright © Software AG 2002

CMPO - Compilation Options

This Natural profile parameter is for mainframes only.

It can be used, at session start, to specify dynamically or to override the same options which you can specify statically with the NTCMPO macro in the parameter module or, during an active session, with the COMPOPT system command.

Possible settings	see COMPOPT system command	
Default setting	see COMPOPT system command	
Dynamic specification	YES	This parameter can only be specified dynamically.
Specification within session	NO	

CMPRINT - Batch Output File

This Natural profile parameter is for UNIX/OpenVMS and Windows. It applies to batch mode only.

It is used for the output report resulting from DISPLAY, PRINT and WRITE statement in a Natural program. In addition, Natural commands from CMSYNIN and INPUT data from CMOBJIN are written to CMPRINT.

Possible settings	any string	**Example for OpenVMS:** CMPRINT = ALF$USER:[home.tmp]batch.out **Example for UNIX:** CMPRINT = $HOME/tmp/batch.out **Example for Windows:** CMPRINT=C:\tmp\out.txt
Default setting	none	
Dynamic specification	YES	
Specification within session	NO	

 Copyright © Software AG 2002

CMPRTnn - Additional Report

This Natural profile parameter is for UNIX/OpenVMS and Windows. It applies to batch mode only.

It is used for additional reports referenced by any Natural program executed during the session. "nn" is a two digit decimal number in the range 01 to 31 corresponding to the report number used in a DISPLAY, PRINT and WRITE statement.

Possible settings	any string	**Example for OpenVMS:** CMPRT07 = ALF$USER:[home.tmp]report7.txt **Example for UNIX:** CMPRT07 = $HOME/tmp/report7.txt **Example for Windows:** CMPRT07=C:\tmp\report7.txt
Default setting	none	
Dynamic specification	YES	
Specification within session	NO	

 CMSYNIN - Batch Input File for Natural Commands and INPUT Data

CMSYNIN - Batch Input File for Natural Commands and INPUT Data

This Natural profile parameter is for UNIX/OpenVMS and Windows. It applies to batch mode only.

CMSYNIN is used for the batch input file. It contains Natural commands and data to be read by INPUT statements during execution of Natural programs (optionally). The number of characters actually processed is restricted to 512 characters per line.

Possible settings	any string	**Example for OpenVMS:** CMSYNIN = ALF$USER:[home.tmp]batch.cmd **Example for UNIX:** CMSYNIN = $HOME/tmp/batch.cmd **Example for Windows:** CMSYNIN=C:\tmp\cmd.txt
Default setting	none	
Dynamic specification	YES	
Specification within session	NO	

 Copyright © Software AG 2002

CMWRKnn - Natural Work Files

This Natural profile parameter is for UNIX/OpenVMS and Windows. It applies to batch mode only.

CMWRKnn is used for Natural work files referenced by any Natural program executed during the session."nn"" is a two digit decimal number in the range 01 to 32 corresponding to the number used in a READ WORK FILE or WRITE WORK FILE statement.

Possible settings	any string	**Example for OpenVMS:** CMWRK05 = ALF$USER:[home.tmp]workfile5.sag **Example for UNIX:** CMWRK05 = $HOME/tmp/workfile5.sag **Example for Windows:** CMWRK05=C:\tmp\workfile5.sag
Default setting	none	
Dynamic specification	YES	
Specification within session	NO	

CO - Compiler Output

This Natural profile parameter is for UNIX/OpenVMS platforms only.

This parameter can be used to deactivate or reactivate the compiler operation-in-progress window wherein, by default, the code is displayed as it is processed by the compiler.

Possible settings	ON	Compiler operation-in-progress window on. This setting is useful in the event of a crash, because the code display will stop at the position affected.
	OFF	Compiler operation-in-progress window off.
Default setting	ON	
Dynamic specification	YES	
Specification within session	NO	

 Copyright © Software AG 2002

COMPR - Set RPC Buffer Compression

This Natural profile parameter is for UNIX/OpenVMS and Windows platforms only.

Note for mainframes:
Use the keyword parameter COMPR of the NTRPC macro for static specification, or the subparameter with the same name of the profile parameter RPC.

This parameter can be used to set the RPC buffer compression. It is effective only, if the automatic Natural RPC execution is used (AUTORPC=ON) and the CALLNAT is executed without a stub. If a stub is used, the compression has already been set during stub generation. For details see Using Compression in the Natural Remote Procedure Call documentation.

COMPR is specified on the client side only.

Possible settings	0	No compression will be performed.
	1	The send buffer contains modifiable fields and output fields and the format buffer. The reply buffer contains modifiable fields and input fields.
	2	Same as COMPR=1, additionally the reply buffer also contains the format buffer.
Default setting	1	
Dynamic specification	YES	
Specification within session	YES	At runtime, this value can be overwritten by the parameter maintenance of the SYSRPC utility.

For further information see the Natural Remote Procedure Call documentation.

COMSERVERID - Determine DCOM Server ID

This Natural profile parameter is for UNIX/OpenVMS and Windows platforms only. For mainframes see the subparameter SERVERID of the profile parameter DCOM.

It can be used to determine the DCOM server name (used by NaturalX).

Possible settings	not specified, or 1-32 characters	DCOM server name.
Default setting	not specified	If COMSERVERID is not specified, the default server name "DEFAULT" is used.
Dynamic specification	YES	
Specification within session	NO	

 Copyright © Software AG 2002

CP - Define Code Page Name

This Natural profile parameter is for:

- mainframes
- UNIX/OpenVMS and Windows

It determines the name of the code page used by the Entire Conversion Service (ECS). Currently, it applies only to the Natural RPC facility when using the transport protocol ACI (that is EntireX Broker).

For more information about ECS and the EntireX Broker refer to the section about Software AG's Internationalization in the EntireX Broker documentation.

Possible settings	1 to 40 characters	Valid code page name of Entire Conversion Services.
Default setting	none	
Dynamic specification	YES	
Specification within session	NO	

CSCPATT - Define RPC Client’s Node Pattern

This Natural profile parameter is for OpenVMS platforms only.

It determines which node names will be used for the CSCI transport protocol. CSCPATT has a lower priority than the profile parameter ACIPATT, it is only effective if the pattern specified for ACIPATT did not match.

You can specify a pattern to match node names used for the CSCI transport protocol together with Natural Remote Procedure Call.

CSCPATT is specified on the client side only. It applies only to nodes which were defined in the server maintenance of the SYSRPC utility (because there the transport protocol cannot be specified).

Possible settings	1-8 characters, including *	Pattern for node name matching. All node names matching this pattern are assumed to be CSCI nodes unless ACIPATT has determined node names as ACI nodes first.
Default setting	*	All node names match the pattern. The CSCI transport protocol is used for all of them unless ACIPATT has determined node names as ACI nodes first.
Dynamic specification	YES	
Specification within session	YES	At runtime, this value can be overwritten by the parameter maintenance of the SYSRPC utility.

For further information see the Natural Remote Procedure Call documentation.

 Copyright © Software AG 2002

CSIZE - Size of Con-nect Buffer Area

This Natural profile parameter is for mainframes only.

It determines the size of the Con-nect buffer area. It only applies if Con-nect is installed.

Possible settings	0 - 128	Buffer size in KB.
Default setting	0	If CSIZE=0 is specified or if the requested space is not available, Con-nect cannot be used.
Dynamic specification	YES	
Specification within session	NO	

See the Con-nect Installation documentation for further information.

CSTATIC - Programs Statically Linked to Natural

This Natural profile parameter is for mainframes only.

With this parameter, you can define a list of entry point names of non-Natural programs which are to be linked together with the Natural parameter module (NATPARM).

Possible settings	list of entry point names	If more than one entry point is specified, each adjacent pair must be separated by a comma and the list must be enclosed within parentheses as shown below: CSTATIC=(PROGRAM1,PROGRAM2,PROGRAM3).
Default setting	none	
Dynamic specification	NO	
Specification within session	NO	

Each entry point specified can be called from a Natural program using a CALL statement.

Modules which have been statically linked can be replaced dynamically by loading them during session initialization, see the profile parameter RCA. Modules which are linked neither statically nor dynamically are loaded dynamically when they are first invoked by a CALL statement.

If you want to link programs to a shared nucleus, you have to define them with the CSTATIC parameter in two parameter modules: one linked to the shared nucleus and the other linked to the environment-dependent nucleus.

For further information, see Statically Linked Non-Natural Programs.

 Copyright © Software AG 2002

CVMIN - Control Variable Modified at Input

This Natural profile parameter is for:

- mainframes
- UNIX/OpenVMS and Windows

It determines whether or not a control variable is to be set to MODIFIED when the setting of the field to which the control variable is attached is overwritten by an *identical* setting. This applies regardless of whether the input was entered manually, read from the stack or supplied in batch mode.

Possible settings	ON	If a field setting is overwritten by the same setting, the corresponding control variable will be set to "MODIFIED".
	OFF	If a field setting is overwritten by the same setting, the corresponding control variable will not be set to "MODIFIED".
Default setting	OFF	
Dynamic specification	YES	
Specification within session	NO	

DATSIZE - Size of Buffer for Local Data

This Natural profile parameter is for mainframes only.

With this parameter you can set the minimum and/or the maximum size of the DATSIZE buffer: DATSIZE=(*min*,*max*). This enables you to limit the DATSIZE buffer space that can be obtained by an application.

Possible settings	min 10 - 1024	Buffer size in KB. If the specified maximum setting is exceeded, the application is terminated and a Natural runtime error 900 (Storage overflow while assigning data area) is output. If no maximum or zero is specified, the buffer size will be increased as required and is only limited by the storage thread or region size.
	max 10 - 1024, or 0	
Default setting	32,0	
Dynamic specification	YES	
Specification within session	NO	

Examples:

1. If you want to set only a minimum setting of 64 KB and no upper limit, specify:
   `DATSIZE=(64,0)`
   or simply `DATSIZE=64`
2. If you want to set only a maximum setting of 512 KB, while using the default setting or leaving the minimum setting as profiled in the NATPARM parameter module, specify:
   `DATSIZE=(,512)`
3. If you want to set a minimum setting of 96 KB and a maximum setting of 128 KB, specify:
   `DATSIZE=(96,128)`

Function of the DATSIZE Buffer

At execution time, the DATSIZE buffer holds the local data used by the Natural main program being executed and the local data of all subordinate objects (except FETCHed programs) invoked by this program.

When you use Natural in a development environment, the minimum DATSIZE required is the default setting (that is, 32 KB). A smaller DATSIZE is only possible when using Natural as a runtime-only environment without any Natural utilities being available.

Calculating the DATSIZE Requirement

The actual DATSIZE requirement can be calculated as follows (refer to the illustration below):

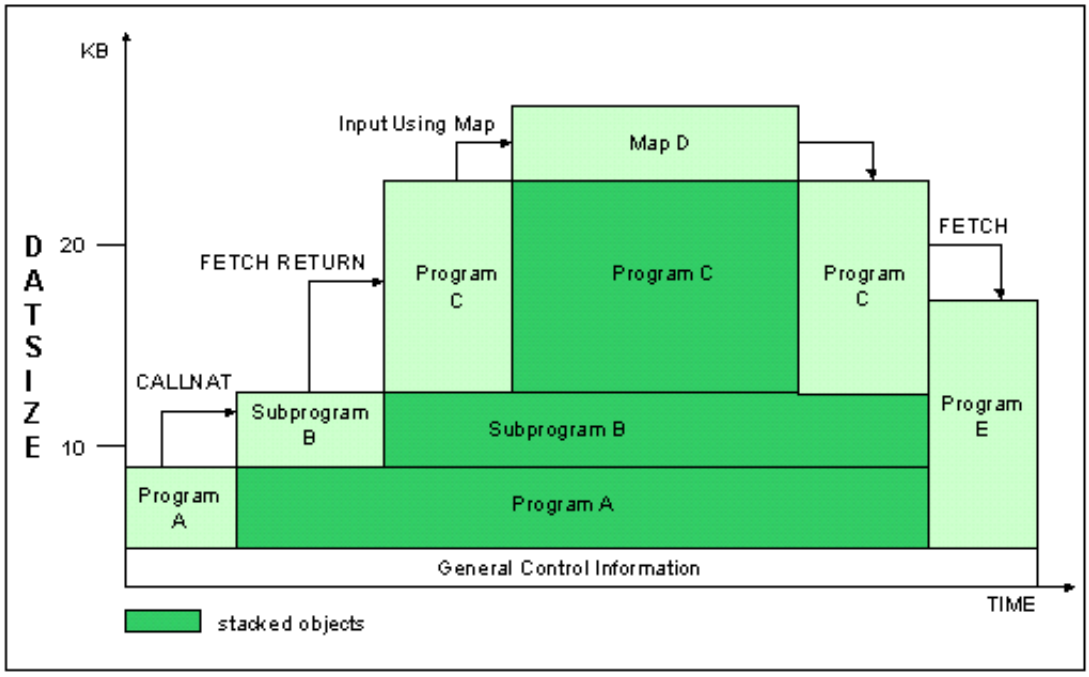
If another object is invoked by the main program, the local data of this object are also held in the DATSIZE buffer.

If other objects are invoked from the invoked object (with a CALLNAT, PERFORM, FETCH RETURN, INPUT USING MAP statement, a helproutine/helpmap being invoked), their local data are also held in the DATSIZE buffer; the local data of an invoked object is held in the DATSIZE buffer until control is returned from the invoked object to the invoking object.

 Copyright © Software AG 2002

If another main program is invoked with a FETCH statement, the local data of all previously invoked objects are deleted from the DATSIZE buffer and the local data of the FETCHed program are held in the DATSIZE.

In addition, an amount of approximately 128 bytes of general control information for execution are held in the DATSIZE buffer, plus approximately 128 bytes of control information for each object whose local data are being held in the DATSIZE buffer. This is illustrated in the figure below.



The **system command** LIST provides an option to display directory information about an object. This information includes the object's DATSIZE storage requirement (not including the control information).

DB - Database Types and Options

This Natural profile parameter is for mainframes only.

It corresponds to the NTDB macro in the parameter module. You can use it to dynamically overwrite specifications made in an NTDB macro.

Possible settings	see NTDB macro	
	OFF	
Default setting	none	
Dynamic specification	YES	This parameter can only be specified dynamically.
Specification within session	NO	

You specify the DB parameter as follows (for one database ID or multiple database IDs):
DB=(*database-type*,*database-id*,*option*,...)
DB=(*database-type*,(*database-id*,...),*option*,...)

You can specify the same database types, database IDs and options as with the NTDB macro.

In addition, you can reset an option dynamically by specifying a corresponding "NO"-prefixed option (for example, NOOPEN, NOREAD).

Moreover, you can specify the option OFF to reset a database definition dynamically to defaults.

If you specify an asterisk (*) as database ID, the specified options apply to all databases for which the default database definition applies (that is, which is not explicitly defined otherwise in an NTDB macro or DB parameter).

Example 1:

```
DB=(VSAM,(22,26,33))
```

This defines Databases 22, 26 and 33 as VSAM databases.

Example 2:

```
DB=(,*,READ)
```

This sets all databases for which the default database definition applies to read-only.

Example 3:

`DB=(,(8,9),NOREAD)` This cancels the read-only option for Databases 8 and 9.

Example 4:

`DB=(,17,OFF)` This resets the database definition of Database 17 to defaults.

Example 5:

```
DB=(ADAV62,(10,15,57),ETP)
```

This defines databases 10, 15 and 57 as Adabas Version 6.2 databases which are to be handled by Entire Transaction Propagator.

 Copyright © Software AG 2002

DB2SIZE - Natural Buffer Area for DB2 or SQL/DS

This Natural profile parameter is for mainframes only. It only applies to Natural for DB2 and Natural for SQL/DS.

It sets the maximum size of the buffer area required by Natural for DB2 and Natural for SQL/DS. If the requested space is not available, the Natural for DB2 or Natural for SQL/DS interface cannot be used.

Possible settings	0 - 64	Maximum size of the buffer area in KB. If Natural is to be used for DB2 or SQL/DS, DB2SIZE must be set to at least 40 KB.
Default setting	0	Set DB2SIZE to 0 if Natural is **not** to be used for DB2 or SQL/DS.
Dynamic specification	YES	
Specification within session	NO	

DBCLOSE - Database Close at Session End

This Natural profile parameter is for mainframes only.

It determines whether or not Natural closes all databases that it has accessed during a session at the end of this session.

Possible settings	ON	Natural closes all databases.
	OFF	Natural does not close any database.
Default setting	OFF	
Dynamic specification	YES	
Specification within session	NO	

 Copyright © Software AG 2002

DBID - Default Database ID of Natural System Files

This Natural profile parameter is for mainframes only.

It identifies the default database in which the Natural system files (FNAT, FUSER, FDIC, FSEC, FSPOOL) are located.

Possible settings	0 - 254, 256 - 65535	Database ID.
Default setting	0	
Dynamic specification	YES	If you specify the DBID parameter dynamically, the database ID for all system files is set to this setting. Therefore, you must specify the DBID parameter **before** any individual system file parameter (FNAT, FUSER, FDIC, FSEC, FSPOOL) if you want to specify any of these parameters, too.
Specification within session	NO	

Database IDs for individual system files can be specified with the parameters FNAT, FUSER, FDIC, FSEC and FSPOOL. The database ID specified with the DBID parameter applies to all Natural system files for which no individual database IDs are specified.

The type of database system is determined by the specification in the NTDB macro.

DBOPEN - Database Open Without ETID

This Natural profile parameter is for mainframes only.

It controls the Open handling of Natural. DBOPEN overrules the setting ETID=blanks.

Possible settings	ON	A database open will be issued even if the ETID parameter is set to blanks.
	OFF	No database open will be issued if the ETID parameter is set to blanks. **Exception**: One open command will always be send to the datebase specified as ETDB, even if ETID is set to blanks and DBOPEN is set to OFF.
Default setting	OFF	
Dynamic specification	YES	
Specification within session	NO	

 Copyright © Software AG 2002

DBROLL - Database Calls Before Roll-Out

This Natural profile parameter is for mainframes and applies under CICS and Com-plete only.

It determines the number of database calls after which a roll-out of the Natural thread will be performed.

Possible settings	0 - 32767	Number of database calls after which a roll-out is performed.
Default setting	0	No roll-out is performed for database calls.
Dynamic specification	YES	
Specification within session	NO	

The Natural session is suspended during the roll-out.

DBSHORT - Interpretation of Database Short Names

This Natural profile parameter is for UNIX/OpenVMS and Windows platforms only.

It can be used to determine whether database short names are to be shown or not.

Possible settings	ON	Short names are shown.
	OFF	Short names are not shown.
Default setting	ON	
Dynamic specification	YES	
Specification within session	NO	

 Copyright © Software AG 2002

DBUPD - Database Updating

This Natural profile parameter is for:

- mainframes
- UNIX/OpenVMS and Windows

It indicates whether database updating is to be permitted during the Natural session.

Possible settings	ON	Database update is permitted.
	OFF	Database update is not permitted. A Natural statement which would cause a database update (STORE, UPDATE, DELETE) is not executed and an error message is generated.
Default setting	ON	
Dynamic specification	YES	
Specification within session	NO	

The DBUPD setting has no effect on the execution of Natural system commands.

DC - Decimal Character

This Natural profile parameter is for:

- mainframes
- UNIX/OpenVMS and Windows

It determines the character to be used for decimal point notation.

Mainframes

Possible settings	any character (except numeric characters)	You specify the DC parameter as DC='*c*' where *c* represents the character to be used as decimal point.
Default setting	. (period)	
Dynamic specification	YES	
Specification within session	YES	Within a Natural session, the profile parameter DC can be overridden by the **session parameter** DC.

The character specified with the DC parameter must not be the same as the one specified with the IA parameter (input assign character) or ID parameter (input delimiter character); it should not be the same as the one specified with the CF parameter (control character for terminal commands) or HI parameter (help character).

UNIX/OpenVMS and Windows

Possible settings	any special character	The character assigned to DC will be in effect for all notations where a decimal point is possible; that is, variables, constants and edit masks.
Default setting	. (period)	
Dynamic specification	YES	
Specification within session	YES	Within a Natural session, the profile parameter DC can be overridden by the **session parameter** DC.

The character specified with the DC parameter must not be the same as the one specified with the IA parameter (input assign character) or ID parameter (input delimiter character); it should not be the same as the one specified with the HI parameter (help character).

Note:
If you change DC in your parameter file, you must resave the DDM you are using in your Natural program which stores a new ".NSD" file on disk.

 Copyright © Software AG 2002

DCOM - NaturalX Support

This Natural profile parameter is for mainframes only.

It corresponds to the NTDCOM macro in the parameter module. It can be used to specify dynamically the same options which you can specify statically in the NTDCOM macro.

Possible settings	See NTDCOM macro.	
Default setting	none	
Dynamic specification	YES	This parameter can only be specified dynamically.
Specification within session	NO	

The entire string of the DCOM parameter has to be enclosed in parentheses, for example:
DCOM=(ACTPOL=EM,AUTOREG=ON,SERVID=’TEST-SERVER’)

DD - Day Differential

This Natural profile parameter is for:

- mainframes
- UNIX/OpenVMS and Windows

It adjusts the current machine date (as read by using the internal machine time) by adding/subtracting any number of days to/from it. This makes it possible to re-run an application that was to be run at a certain date but for some reason could not be run at that date.

The parameter is specified as "DD=+*nn*" or "DD=-*nn*" (*nn* being the number of days).

Mainframes

Possible settings	-32767 to +32767	Machine date is adjusted. Specification of "+" is optional.
Default setting	0	No adjustment is made.
Dynamic specification	YES	
Specification within session	NO	

See also the parameters TD and YD.

UNIX/OpenVMS and Windows

Possible settings	-10953 to +10953	Machine date is adjusted. Specification of "+" is optional.
Default setting	0	No adjustment is made.
Dynamic specification	YES	
Specification within session	NO	

 Copyright © Software AG 2002

DELETE - Deletion of Dynamically Loaded Programs

This Natural profile parameter is for mainframes only.

It determines whether dynamically loaded non-Natural programs are to be deleted at the completion of the Natural program in which they are used.

Possible settings	ON	Dynamically loaded non-Natural programs are deleted at the end of the Natural program in which they were loaded.
	OFF	Dynamically loaded non-Natural programs are not deleted at the end of the Natural program in which they were loaded; they are kept until command mode is entered again, and the deletion is performed immediately before Natural enters command mode processing.
Default setting	ON	
Dynamic specification	YES	
Specification within session	NO	

The following platform-specific requirements apply:

Platform:	**Comment:**
Under CICS	In a CICS environment, this parameter applies only the non-Natural program is invoked via standard linkage conventions (SET CONTROL ’P=S’).
Under OS/390 Batch, TSO, VSE/ESA Batch and IMS/TM	This parameter does not apply in an IBM Language Environment (LE). All dynamic subprograms loaded during a Natural session are deleted upon LE environment termination, that is, during the termination of the Natural session. For more information about Natural in an LE environment, see Support of IBM Language Environment Subprograms.

DFOUT - Date Format for Output

This Natural profile parameter is for:

- mainframes
- UNIX/OpenVMS and Windows

It determines the format in which the settings of date variables are displayed by INPUT, DISPLAY, PRINT and WRITE statements.

Possible settings	S	Date variables are displayed with a 2-digit year component, and delimiters as determined by the profile parameter DTFORM. For example: *yy-mm-dd*.
	I	Date variables are displayed with a full 4-digit year component and no delimiters. For example: *yyyymmdd*.
Default setting	S	
Dynamic specification	YES	
Specification within session	YES	Within a Natural session, the profile parameter DFOUT can be overridden by the **session parameter** DFOUT.

The DFOUT parameter is evaluated at runtime. It applies to date fields in INPUT, DISPLAY, PRINT and WRITE statements for which no explicit edit mask is specified and for which the **session parameter** DF is not set.

For either DFOUT setting, the sequence of the day, month and year components in the date settings is determined by the DTFORM profile parameter.

See also the section Processing of Date Information in the Natural Programming Guide documentation.

 Copyright © Software AG 2002

DFS - Specify RPC Client’s Default Server Address

This Natural profile parameter is for UNIX/OpenVMS and Windows platforms only.

Note for mainframes:
Use the keyword parameter DFS of the NTRPC macro for static specification, or the subparameter with the same name of the profile parameter RPC.

This parameter can be used to define an RPC default server address. It determines the server name, the server node, the logon indicator and the transport protocol. The default server address will be used only if no appropriate server is found in the service directory. For further information, see Specifying RPC Server Adresses in the Natural Remote Procedure Call documentation.

DFS is specified on the client side only.

To define a default server address you specify up to 4 subparameters.

Possible settings	4 subparameters	The *server name* (1 - 8 characters). There is no default, the value must be specified.
		The *server node* (1 - 8 characters). There is no default, the value must be specified.
		A *logon indicator*. If nothing is specified, blank is the default. L The client initiates a Natural logon to the server with the library name of the current library on the client. blank No server logon will be executed. Note for Windows platforms: instead of "L", check the selection box.
		The *transport protocol* to be used (ACI or CSCI). ACI is the default. CSCI is supported only for OpenVMS.
Default setting	none	Subparameter defaults, see above.
Dynamic specification	YES	See below.
Specification within session	YES	At runtime, this value can be overwritten by a Natural user exit (USR2007N).

For dynamic specification the syntax is as follows:
DFS=(*server-name*,*server-node name*,*logon-indicator*,*transport-protocol-name*)

For further information see the Natural Remote Procedure Call documentation.

DFSTACK - Date Format for Stack

This Natural profile parameter is for:

- mainframes
- UNIX/OpenVMS and Windows

It determines the format in which the settings of date variables are placed on the stack via a STACK, RUN or FETCH statement.

Possible settings	C	Same as DFSTACK=S. In addition, if the century used when the setting is read from the stack is not the same as that of the original date setting, Natural will issue a runtime error.
	I	Date variables are placed on the stack with a full 4-digit year component and no delimiters. For example: *yyyymmdd*.
	S	Date variables are placed on the stack with a 2-digit year component, and delimiters as determined by the profile parameter DTFORM. For example: *yy-mm-dd*.
Default setting	S	
Dynamic specification	YES	
Specification within session	YES	Within a Natural session, the profile parameter DFSTACK can be overridden by the **session parameter** DFSTACK.

The DFSTACK parameter does not apply to STACK, RUN or FETCH statements for which the **session parameter** DF is set.

See also the section Processing of Date Information in the Natural Programming Guide.

 Copyright © Software AG 2002

DFTITLE - Output Format of Date in Standard Report Title

This Natural profile parameter is for:

- mainframes
- UNIX/OpenVMS and Windows

It determines the output format of the date in the default title line of a report page (as output with a DISPLAY, WRITE or PRINT statement). DFTITLE is evaluated at runtime and determines whether the date is displayed with a 2-digit or 4-digit year component with or without delimiters. It has no effect on a user-defined page title (as specified with a WRITE TITLE statement).

Possible settings	S	The date is output with a 2-digit year component and delimiters. For example: *yy-mm-dd*.
	L	The date is output with a 4-digit year component and delimiters. For example: *yyyy-mm-dd*.
	I	The date is output with a 4-digit year component and no delimiters. For example: *yyyymmdd*.
Default setting	S	
Dynamic specification	YES	
Specification within session	YES	Within a Natural session, the profile parameter DFTITLE can be overridden by the **session parameter** DFTITLE.

The sequence of the day, month and year components and the delimiter characters used are determined by the profile parameter DTFORM.

See also Processing of Date Information in the Natural Programming Guide.

DLISIZE - Size of Natural Buffer Area for DL/I

This Natural profile parameter is for mainframes only. It only applies to Natural for DL/I.

It determines the maximum size of the buffer area required by Natural for DL/I. If the requested space is not available, Natural for DL/I cannot be used.

Possible settings	0, or 26-512	Buffer size in KB. The size actually required depends on the specifications in the NDLPARM macro (see the Natural for DL/I documentation). If you use the default specifications in NDLPARM, DLISIZE=26 is sufficient.
Default setting	0	If you do not need DL/I support during a Natural session, you are recommended to invoke Natural with DLISIZE=0 to avoid overhead caused by handling of unused buffers.
Dynamic specification	YES	
Specification within session	NO	

If the size specified with the DLISIZE parameter is not sufficient, an appropriate error message at initialization of Natural for DL/I will tell you what size to specify.

If Natural for DL/I is installed, the corresponding Natural buffers are requested at the initialization of the Natural session.

 Copyright © Software AG 2002

DSC - Data-Stream Compression (for 3270-Type Terminals)

This Natural profile parameter is for mainframes only. It only applies to 3270-type terminals.

With this parameter, you can switch off Natural's automatic optimization of the screen data stream for 3270-type terminals.

Possible settings	ON	Data-stream compression is used.
	OFF	Data-stream compression is not used.
Default setting	ON	
Dynamic specification	YES	
Specification within session	NO	

Natural's screen optimization causes screen data to be sent as compressed as possible. If this should conflict with any TP monitor's screen optimization or hardware limitation, you can use this parameter to switch off Natural's screen optimization; screen data will then be sent in non-compressed form.

This parameter has the same function as the **terminal command** %RO described in the Natural Reference documentation.

If you use the BX session parameter settings BX=L or BX=R, you should switch off Natural's screen optimization using DSC=OFF.

Copyright © Software AG 2002

DSIZE - Size of Debug Buffer Area

This Natural profile parameter is for mainframes only.

It specifies the size of the Natural debug buffer area. If DSIZE is set to 0 or if the requested space is not available, the DBLOG function cannot be used.

Possible settings	0, or 2 - 64	Buffer area in KB.
Default setting	2	
Dynamic specification	YES	
Specification within session	NO	

This buffer area is used by the DBLOG function which is described in the section Debugging and Monitoring.

 Copyright © Software AG 2002

DSLM - Size Limitation

This Natural profile parameter is for UNIX/OpenVMS and Windows platforms only.

It limits index values, number of array occurrences and array sizes to the limits applicable for Natural Version 3.1. for mainframe computers. The parameter is used for compatibility reasons only.

Possible settings	ON	The above mentioned items will be limited.
	OFF	No limit is in effect.
Default setting	OFF	
Dynamic specification	YES	
Specification within session	NO	

DTFORM - Date Format

This Natural profile parameter is for:

- mainframes
- UNIX/OpenVMS and Windows

It indicates the default format in which dates are to be provided automatically by Natural as part of the default title on Natural reports, as date constants and date input.

Possible settings	E, G, I,U	**Setting (Area)**	**Date Format**
		E (Europe)	DD/MM/YYYY
		G (Germany)	DD.MM.YYYY
		I (International)	YYYY-MM-DD
Default setting	I		
Dynamic specification	YES		
Specification within session	NO		

The first day of a week is assumed to be Monday - unless DTFORM=U is specified, in which case Sunday is used.

For date constants, the year component (YYYY) consists of all four digits. Only the last two digits of the year component are used for reports, date input, the Natural system function VAL, and when the date is moved to an alphanumeric field.

The output format of the date in a default report page title is also specified by the profile parameter DFTITLE.

 Copyright © Software AG 2002

DU - Dump Generation

This Natural profile parameter is for:

- mainframes
- UNIX/OpenVMS and Windows

Mainframes

This parameter indicates whether a memory dump is to be generated in the case of an abnormal termination during the Natural session.

Possible settings	ON	A memory dump is produced in the case of an abnormal termination (TP-monitor dump dataset or SYSUDUMP in OS/390 batch mode or TSO). Then the Natural session terminates with error message NAT9974.
	OFF	No memory dump is produced. In batch mode, subsequent action taken by Natural is determined by the setting of the CC profile parameter. In online mode, Natural responds with errors NAT0953, NAT0954, NAT0955 or NAT0956. For further information on the abnormal termination, you can use the system command DUMP.
	SNAP	This will force an immediate dump in the case of an abnormal termination during a Natural session. The Natural session will continue after the dump has been taken.
	FORCE	This will force an immediate dump in the case of an abnormal termination during a Natural session and will terminate the Natural session immediately. This is useful for testing purposes in some environments.
Default setting	OFF	
Dynamic specification	YES	
Specification within session	YES	Within a Natural session, the profile parameter DU can be overridden by the **session parameter** DU.

DU=ON, SNAP or FORCE may make system performance considerably worse, due to I/O processing on the dump dataset.

UNIX/OpenVMS and Windows

This parameter indicates whether a disassembled object code dump is to be generated.

Possible settings	ON	When the system command CHECK, STOW, CATALOG, RUN or CATALL is executed, a disassembled object code file is produced. This dump file is written into the directory "tmp" below the Natural root directory. The name of this dump file consists of the source file name and the extension ".DIA". If the source file has not been saved, the name of the dump file is "GEN.DIA". If the program contains database access statements, dump files with the extension ".ADA" (for Adabas) or ".SQL" (for SQL databases) are also created. If XREF data are generated, a dump file ".xrf" is created. **Note:** DU=ON may create a large dump file (depending on the size of the source file), which can cause significant degradation in system performance.
	OFF	No dump file is generated.
Default setting	OFF	
Dynamic specification	YES	
Specification within session	YES	Within a Natural session, the profile parameter DU can be overridden by the **session parameter** DU.

 Copyright © Software AG 2002

DYNPARM - Control Use of Dynamic Parameters

This Natural profile parameter is for:

- mainframes
- UNIX/OpenVMS and Windows

Mainframes

This parameter can only be specified dynamically and can be used only once.
It controls the use of dynamic profile parameters. It corresponds to the NTDYNP macro in the parameter module.

Possible settings	ON	All profile parameters can be specified dynamically.
	OFF	No profile parameters can be specified dynamically.
	DYNPARM=(ON,*parameter-name*,...)	Only those parameters whose *parameter-names* are specified here can be specified dynamically. Other parameters cause Error Message NAT7008 to be issued.
	DYNPARM=(OFF,*parameter-name*,...)	All profile parameters can be specified dynamically - except those whose *parameter-names* are specified here. These parameters cause Error Message NAT7008 to be issued.
Default setting	ON	
Dynamic specification	YES	The DYNPARM parameter can only be used if NTDYNP is not specified in the Natural parameter module.
Specification within session	NO	

The parameter restrictions defined by DYNPARM (or NTDYNP) do not apply within PROFILE or SYS parameter strings, if defined outside of these strings. If DYNPARM is used within PROFILE or SYS strings, it replaces any previous restrictions defined by DYNPARM or macro NTDYNP outside of PROFILE or SYS strings.

DYNPARM can be used only once within one string and should be placed at the end of it.

Example:

This example illustrates restricting of the dynamic parameters FNAT and FSEC. In NATPARM the following parameter restriction should be defined:

```
NTPRM DBID=0,FNR=0
NTDYNP ON,PROFILE
```

Additionally, almost all parameter profiles could look like the following:

```
...,FNAT=(22,7,PASSW),FSEC=(22,9,PASSW),DYNPARM=(OFF,FNAT,FSEC)
```

If some special users are to be allowed to use all parameters including FNAT and FSEC, their parameter profiles could look like the following:

```
USER=(ADM1,ADM2),...,FNAT=(22,8),FUSER=(22,12),DYNPARM=(OFF,DUMMY)
```

This example forces normal users to enter the PROFILE parameter as the first dynamic parameter. Subsequently all parameters except FNAT and FSEC are allowed. Of course the access to the parameter profile application SYSPARM must be restricted.

 Copyright © Software AG 2002

UNIX/OpenVMS and Windows

This parameter enables/disables the use of Natural dynamic parameters.

	ON	Dynamic parameters supplied during Natural startup are processed.
Possible settings	OFF	Dynamic parameters supplied during Natural startup are not processed. **Note:** If DYNPARM is set to OFF in the Natural default parameter file NATPARM, no alternative user-defined parameter files can be used when starting Natural.
Default setting	ON	
Dynamic specification	NO	
Specification within session	NO	

See Dynamic Assignment of Parameter Values for additional information on the use of dynamic parameters.

ECHO - Control Printing of Batch Input Data

This Natural profile parameter is for:

- mainframes
- UNIX/OpenVMS and Windows

It only applies in batch mode. ECHO is used to enable or disable the printing of input data provided to Natural during batch mode processing.

Possible settings	ON	Natural prints the input data provided during batch mode processing to the batch output file CMPRINT.
	OFF	Natural does *not* print input data provided during batch processing.
Default setting	ON	
Dynamic specification	YES	
Specification within session	NO	

It is also possible to suppress printing of a *single input line* by preceding it with a line containing the terminal command for record suppression %*.

 Copyright © Software AG 2002

ECPMOD - Entire Connection Protocol Mode

This Natural profile parameter is for UNIX/OpenVMS platforms only.

It determines the protocol that is used when downloading workfiles.

Possible settings	ON	The tcpip protocol is used.
	OFF	The zmodem protocol is used.
Default setting	ON	
Dynamic specification	YES	
Specification within session	NO	

EDITOR - Other Program Editor

This Natural profile parameter is for UNIX/OpenVMS only.

It is modified to invoke a program editor other than the standard editor provided by Software AG.

Possible settings	any character string	Name of the editor you want to use. Specification, see below. **Note:** You can still use the former Natural program editor (as used with Natural versions prior to Version 2.2) by specifying NATCEDIT.
Default setting	NATEDIT	Standard editor provided by Software AG.
Dynamic specification	NO	
Specification within session	NO	

To use an external editor specify the path and editor name:

***path-name editor-name command-to-specify-line* %l *command-to-specify-file* %f**

The string before **%l** is replaced by the ASCII representation of the line number. The string before **%f** is replaced by the file name as known by Natural.

Examples for OpenVMS:

$ EDIT/TPU %f
$ EDIT/TPU/COM = *mydir* :NATKEY.TPU %f
$ EDIT/TPU /START_POSITION = %l %f

A sample TPU file named NATKEY.TPU is located in the Natural profile directory (PROFILE_PATH).

Example for UNIX:

vi +%l %f

To use "vi" as a program editor, edit your parameter files as specified. If the existing parameter files are not updated, the specified editor (that is, "vi") cannot find the error line.

 Copyright © Software AG 2002

EDPSIZE - Size of Software AG Editor Auxiliary Buffer Pool

This Natural profile parameter is for mainframes only.

It determines the size of the auxiliary Software AG Editor buffer pool. It must be used when the Software AG Editor runs in a Sysplex environment. It allows the Software AG Editor to be run without a Software AG Editor (local or global) buffer pool.

Possible settings	0, or 32 - 32767	Buffer size in KB. If EDPSIZE is not zero, the auxiliary buffer pool is used although a (local or global) Software AG Editor buffer pool is defined with the NTBPI macro or BPI profile parameter.
Default setting	0	
Dynamic specification	YES	
Specification within session	NO	

No Software AG Editor work file is required for the auxiliary buffer pool.

When the auxiliary buffer pool is used, the Software AG Editor's recovery function is not available.

The auxiliary buffer pool storage is allocated exclusively for the user session. This means, that sufficient space in the user thread or in the region must be available.

For further information on the Software AG Editor, see Operating the Software AG Editor (in the Natural Operations for Mainframes documentation).

EDTBPSIZE - Software AG Editor Buffer Pool Size

This Natural profile parameter is for UNIX/OpenVMS and Windows platforms only.

It sets the size of the Software AG Editor buffer pool (in KB).

Possible settings	0 - 4000	Size of the buffer pool in KB.
Default setting	400	
Dynamic specification	NO	
Specification within session	NO	

 Copyright © Software AG 2002

EDTLFILES - Number of Software AG Editor Logical Files

This Natural profile parameter is for UNIX/OpenVMS and Windows platforms only.

It sets the maximum number of the Software AG Editor sessions a user can open at a time.

Possible settings	10 - 999	Maximum number of SAG Editor sessions.
Default setting	100	
Dynamic specification	NO	
Specification within session	NO	

EDTRB - Program Editor Ring Buffer

This Natural profile parameter is for UNIX/OpenVMS platforms only.

It can be used to determine whether the ring buffer of the program editor is to be used or not.

For further information, see the section Multiple Editor Session.

Possible settings	ON	Ring buffer is used.
	OFF	Ring buffer is not used.
Default setting	OFF	
Dynamic specification	NO	
Specification within session	NO	

 Copyright © Software AG 2002

EJ - Page Eject

This Natural profile parameter is for:

- mainframes
- UNIX/OpenVMS and Windows

It determines whether a page eject is to be performed as a result of a logical page break, a break between program input and output, and the "normal end" message.

Possible settings	ON	A page eject is performed.
	OFF	No page eject is performed.
Default setting	ON	
Dynamic specification	YES	
Specification within session	YES	Within a Natural session, the profile parameter EJ can be overridden by the **session parameter** EJ. The EJ setting can in turn be overridden by an EJECT statement.

This parameter only applies to the first report (report 0). For additional reports, the **statement** EJECT with report specification (rep) has to be used.

ENDBT - BACKOUT TRANSACTION at Session End

This Natural profile parameter is for mainframes only.

It determines whether or not an implicit BACKOUT TRANSACTION statement is to be issued at the end of the Natural session.

Possible settings	ON	Natural will issue an implicit BACKOUT TRANSACTION statement at session end.
	OFF	Natural will not issue an implicit BACKOUT TRANSACTION statement at session end.
Default setting	ON	
Dynamic specification	YES	
Specification within session	NO	

 Copyright © Software AG 2002

ENDMSG - Display Session-End Message

This Natural profile parameter is for:

- mainframes
- UNIX/OpenVMS and Windows

Mainframes

By default, message NAT9995 is displayed at the end of the Natural session, indicating that the Natural session has been ended normally. With this parameter, you can suppress the display of this message.

Possible settings	ON	Message NAT9995 will be displayed at the end of the session.
	OFF	Message NAT9995 will not be displayed at the end of the session.
Default setting	ON	
Dynamic specification	YES	
Specification within session	NO	

If a session back-end program is defined (with the profile parameter PROGRAM, the ENDMSG parameter has no effect; the message text will then be passed to the back-end program in the parameter area and will not be displayed by Natural.

UNIX/OpenVMS and Windows

This parameter applies to batch mode only.

By default, message NAT9995 is displayed at the end of the Natural session, indicating that the Natural session has been ended normally. With this parameter, you can suppress the display of this message.

Possible settings	ON	Message NAT9995, NAT9978 or NAT9987 is written to the batch output file CMPRINT at the end of the session.
	OFF	Message NAT9995, NAT9978 or NAT9987 does not appear in CMPRINT.
Default setting	ON	
Dynamic specification	YES	
Specification within session	NO	

If Natural terminates with a startup error, then Natural message NAT9987 will be used instead of NAT9995.

If Natural terminates with a run-time error, then Natural message NAT9978 will be used instead of NAT9995.

 Copyright © Software AG 2002

ENDIAN - Endian Mode for Compiled Objects

This Natural profile parameter is for UNIX/OpenVMS and Windows only.

It specifies the architecture for which the compiler should generate GP. See also Portable Natural Generated Programs (in the Natural Programming Guide).

Possible settings	DEFAULT	Endian mode is derived from the architecture currently used.
	LITTLE	The compiler generates GP for Little Endian mode.
	BIG	The compiler generates GP for Big Endian mode.
Default setting	DEFAULT	
Dynamic specification	YES	
Specification within session	YES	Within a Natural session, the profile parameter settings can be overwritten by the session parameter ENDIAN.

ESCAPE - Ignore Terminal Commands %% and %.

This Natural profile parameter is for:

- mainframes
- UNIX/OpenVMS and Windows

It can be used to disable the **terminal commands** " %%" and "%.".

Possible settings	ON	The terminal commands "%%" and "%." can be used.
	OFF	The terminal command "%%" and "%." will be ignored; that is, it will not be possible to leave the currently active Natural program or the Natural session respectively by entering "%%" or "%.".
Default setting	ON	
Dynamic specification	YES	
Specification within session	NO	

 Copyright © Software AG 2002

ESIZE - Size of User-Buffer Extension Area

This Natural profile parameter is for mainframes only.

It sets the size of the user-buffer extension area. This parameter determines the size (in KB) of the Natural source area which is used by the Natural editors. Note that you can only set a setting smaller than the default setting in a runtime environment (where the editors are not used).

Possible settings	2 - 512	Buffer extension area in KB.
Default setting	28	
Dynamic specification	YES	
Specification within session	NO	

The user-buffer extension area contains:

- the source code of the Natural programming object to be compiled,
- the table of currently active PA/PF keys,
- the Natural stack area,
- the currently active global data area,
- other tables and work areas internally used by Natural.

In a production environment, Natural sources are not needed and the ESIZE can therefore be reduced accordingly.

If this area is not large enough to contain the necessary information, Error Message NAT0886 is issued.

 ESXDB - Database ID used for Entire System Server DDMs

ESXDB - Database ID used for Entire System Server DDMs

This Natural profile parameter is for Unix/OpenVMS and Windows platforms only and applies to the Entire System Server Interface.

It specifies the database ID used for Entire System Server's DDMs.

Possible settings	0 or 1-254	Database ID. To activate this parameter, a database ID in the range of 1 to 254 must be specified. **Note:** Entire System Server's DDMs are cataloged with DBID=148. If you are using an Adabas database with this DBID, specify a different number for ESXDB. For information on how to do this, please refer to Setting up the Entire System Server Interface Setting up the Entire System Server Interface Setting up the Entire System Server Interface
Default setting	0	ESXDB is not active.
Dynamic specification	NO	
Specification within session	NO	

 Copyright © Software AG 2002

ET - Execution of END/BACKOUT TRANSACTION Statements

This Natural profile parameter is for:

- mainframes
- UNIX/OpenVMS and Windows

It specifies for which databases END TRANSACTION and BACKOUT TRANSACTION statements are to be executed.

Possible settings	ON	END TRANSACTION and BACKOUT TRANSACTION statements are executed for all databases which have been referenced since the beginning of the Natural session or since the last execution of an END TRANSACTION and BACKOUT TRANSACTION statement.
	OFF	END TRANSACTION and BACKOUT TRANSACTION statements are executed only for the databases affected by the transaction (and - if applicable - for the database to which transaction data are written).
Default setting	OFF	
Dynamic specification	YES	
Specification within session	NO	

Any updates to a database which are not executed under the control of Natural (that is, by native invocation of the database link routines) do not affect the Natural transaction logic.

ETA - Error Transaction Program

This Natural profile parameter is for:

- mainframes
- UNIX/OpenVMS and Windows

It provides the name of the program which receives control if an error condition is detected during Natural program execution.

Possible settings	1 to 8 characters	Program name for error transaction.
Default setting	none	No error transaction program is called.
Dynamic specification	YES	
Specification within session	YES	

The setting of this parameter can be modified by a user program with the system variable *ERROR-TA.

Note:
Error transaction programs must be in the library to which you are currently logged on.

When an error occurs, Natural executes a STACK TOP DATA statement and places the following information at the top of the stack:

- Error number (N4 if SG=OFF; N5 if SG=ON)
- Line number (N4)
- Status (A1)
- Program name (A8)
- Level (N2)

This information can be used as INPUT data by an error transaction. The status can be one of the following settings:

Setting	Explanation
C	Command processing error
L	Logon processing error
R	Error on Remote server (in conjunction with Natural RPC)
O	Object time error
S	Non-correctable Syntax error

If the status is either "C" or "L", the line number will be "0".

If **Natural Security** is installed, the following rules apply:

- If an error occurs during the *first* logon to Natural, the program specified with the ETA parameter applies. The error transaction must be in SYSLIB at the time of the first logon.
- If an error occurs *after* the first logon, the program specified as error transaction in the security profile of the current library applies. If no error transaction is specified, the ETA parameter applies.

 Copyright © Software AG 2002

For additional information concerning the definition and use of error transaction programs, see Transactions (in the Natural Security documentation).

ETDB - Database for Transaction Data

This Natural profile parameter is for:

- mainframes
- UNIX/OpenVMS and Windows

It specifies the database in which transaction data, as supplied with an END TRANSACTION statement is to be stored.

Possible settings	0 - 254, 256 - 65535	Database number.
Default setting	0	
Dynamic specification	YES	
Specification within session	NO	

Mainframes

If the ETDB parameter is not specified, the transaction data are written to the database where the Natural Security system file (FSEC) is located.

If FSEC is not specified, it is considered to be identical to the Natural system file FNAT (if Natural Security is not installed, the transaction data are written to the database where FNAT is located).

UNIX/OpenVMS and Windows

If the ETDB parameter is not specified, the transaction data is written to the database specified with the parameter UDB.

 Copyright © Software AG 2002

ETEOP - Issue END TRANSACTION at End of Program

This Natural profile parameter is for mainframes only.

It determines whether or not an implicit END TRANSACTION statement is to be issued at the end of a Natural program (that is, before NEXT mode is reached).

Possible settings	ON	Natural will issue an implicit END TRANSACTION statement at the end of a Natural program.
	OFF	Natural will not issue any implicit END TRANSACTION statement at the end of a Natural program.
Default setting	ON	
Dynamic specification	YES	
Specification within session	NO	

ETID - Adabas User Identification

This Natural profile parameter is for:

- mainframes
- UNIX/OpenVMS and Windows

It is used as an identifier for Adabas-related information; for example, for identification of data stored as a result of an END TRANSACTION statement.

Mainframes

Possible settings	1 to 8 characters	The setting is used as the user ID setting in an Adabas open call.
	OFF	Natural does not issue any Adabas open and close commands at the beginning of the Natural session. If, however, any ETID and/or OPRB specifications are present in Natural Security, these specifications are used in the subsequent open issued by Natural Security. This parameter setting is provided for use in conjunction with Natural Security to prevent Natural batch jobs that are sent at the same time from causing duplicate user ID settings in an Adabas open call during the initialization phase.
Default setting	*INIT-USER	
Dynamic specification	YES	
Specification within session	NO	

If the ETID parameter is set to blanks, Natural does not issue any Adabas open and close commands; the OPRB parameter (if specified) and any ETID and OPRB specifications in Natural Security are ignored.
In this case, you are recommended to set the Natural profile parameter DBCLOSE to ON to enforce a close command at session end. Otherwise, the user is not logged off from Adabas and the Adabas user queue element is not deleted. This may cause an overflow situation in the Adabas user queue.

If the ETID setting is **not** the same as the setting of the Natural system variable *INIT-USER, Natural issues an Adabas open with the specified ETID setting (and OPRB setting, if specified) at the beginning of the Natural session; this open remains in effect until the end of the Natural session; any ETID and OPRB specifications in Natural Security are ignored.

If the ETID setting is the same as the setting of *INIT-USER, or if the ETID parameter is not specified, Natural issues an Adabas open with the *INIT-USER setting as ETID (and the OPRB setting, if specified) at the beginning of the Natural session. If any Natural Security logon (initial logon or any subsequent logon) would change the currently valid ETID or OPRB setting (due to the library-/user-specific ETID and OPRB specifications in Natural Security), Natural Security issues a new open with the new ETID and OPRB settings. If the settings would remain the same after a logon, Natural Security does not issue a new open.

ETID and *INIT-USER can be modified by user exit NATUEX1 during session startup. See NATUEX1 - User Exit for Authorization Control (in the Natural Operations for Mainframes documentation).

 Copyright © Software AG 2002

ETID - UNIX/OpenVMS and Windows

Possible settings	1 to 8 characters	The setting is used as the user ID setting in an Adabas open call.
	OFF	
Default setting	none	
Dynamic specification	YES	
Specification within session	NO	

The Adabas user ID has its own syntax. Please consult your Adabas Command Reference Documentation if you want to use special characters for the setting of ETID.

If the ETID is not specified neither in NATPARM nor dynamically, Natural uses the setting of *INIT-USER to fill the ETID. Thus, the only way to have no (or rather a blank) user ID passed to Adabas C with each call, you need to specify dynamically ETID=" ".

Running under UNIX, you can dynamically specify "ETID=$$". The UNIX shell will replace $$ by the process id of your shell process before executing Natural. This way you set a virtually unique ETID.

The ETID might also be used by Natural for work-file name generation. This might cause problems when running multiple instances on one machine or again, if special characters are used for the setting of ETID. Please refer to the Natural parameter TMPSORTUNIQ for a solution.

ETIO - Issue END TRANSACTION upon Terminal I/O

This Natural profile parameter is for mainframes only.

It determines whether or not implicit END TRANSACTION statements are to be issued upon terminal I/Os.

Possible settings	ON	Natural will issue an implicit END TRANSACTION statement whenever a terminal I/O occurs. Whenever a transaction monitor commits the associated databases because of a terminal I/O, all related databases are also committed. This is useful for the synchronization of database transactions.
	OFF	Natural will issue no implicit END TRANSACTION statements upon terminal I/Os.
Default setting	ON	
Dynamic specification	YES	
Specification within session	NO	

 Copyright © Software AG 2002

ETPDB - Databases Containing Entire Transaction Propagator Master Files

This Natural profile parameter is for UNIX/OpenVMS and Windows platforms and only applies to Software AG's Entire Transaction Propagator.

ETPDB specifies the databases that contain Entire Transaction Propagator master files.

Possible settings	0-255	Database IDs.
Default setting	none	
Dynamic specification	NO	
Specification within session	NO	

Note for UNIX users:
Mark all databases in the parameter's database list that contain master files; for details see the Entire Transaction Propagator documentation.

Note for Windows users:
The entries must be separated by blanks.

ETPSIZE - Size of Entire Transaction Propagator Buffer

This Natural profile parameter is for:

- mainframes
- UNIX/OpenVMS and Windows

This parameter only applies if Entire Transaction Propagator is installed. It determines the size of the Entire Transaction Propagator buffer.

Mainframes

Possible settings	0, or 10 - 128	Size of the Entire Transaction Propagator buffer in KB. If Entire Transaction Propagator is to be used, a setting has to be specified for this parameter; see the Entire Transaction Propagator documentation.
Default setting	0	Entire Transaction Propagator cannot be used.
Dynamic specification	YES	
Specification within session	NO	

UNIX/OpenVMS and Windows

Possible settings	0, 10 - 63	Size of the Entire Transaction Propagator buffer in KB. If Entire Transaction Propagator is to be used, a setting has to be specified for this parameter; see the Entire Transaction Propagator documentation. If Entire Transaction Propagator is not to be used, set ETPSIZE to 0.
Default setting	0	If ETPSIZE=0 or if the requested space is not available, the Entire Transaction Propagator cannot be used.
Dynamic specification	NO	
Specification within session	NO	

 Copyright © Software AG 2002

ETRACE - External Trace Function

This Natural profile parameter is for mainframes only.

It is used to activate/deactivate the external trace function.

Do not use this parameter without prior consultation of Software AG Support.

The trace function is intended primarily for Software AG internal use for debugging purposes. It writes trace data to an external trace dataset depending on the TP environment in which Natural is running. In batch and TSO environments, a dataset (CMTRACE) is required for the external trace.

Possible settings	ON	Activates the (normal) external trace function.
	OFF	Deactivates the (normal) external trace function.
Default setting	OFF	
Dynamic specification	YES	
Specification within session	YES	Within a Natural session, the **terminal command** %TRE can be used to activate/deactivate the external trace function, except GTF.

Using GTF

ETRACE=(ON,GTF) **ETRACE=(OFF,GTF)**	Activates/deactivates the Generalized Trace Facility (GTF) offered by OS/390. The trace records are written to the GTF. This is possible under OS/390 and TSO.
ETRACE=(ON,NOGTF) **ETRACE=(OFF,NOGTF)**	Activates/deactivates the (normal) external trace function.
ETRACE=(,GTF)	Equivalent to ETRACE=GTF. Trace data is written to the GTF. ON or OFF is not altered.

EXCSIZE - Size of Buffer for Natural Expert C Interface

This Natural profile parameter is for mainframes only.

It determines the size of the buffer required by the C interface of Natural Expert. See the Natural Expert documentation.

Possible settings	0 - 256	Buffer size in KB.
Default setting	0	
Dynamic specification	YES	
Specification within session	NO	

 Copyright © Software AG 2002

EXRSIZE - Size of Buffer for Natural Expert Rule Tables

This Natural profile parameter is for mainframes only.

It determines the size for the buffer required by the rule tables of Natural Expert. See the Natural Expert documentation.

Possible settings	0 - 256	Buffer size in KB.
Default setting	0	
Dynamic specification	YES	
Specification within session	NO	

 FAMSTD - Overwriting of Print and Work File Access Method Assignments

FAMSTD - Overwriting of Print and Work File Access Method Assignments

This Natural profile parameter is for mainframes only.

It controls the automatic overwriting of print and work file access method assignments during session initialization according to the dataset definition in the job control.

See also the AM subparameter of the macros NTPRINT and NTWORK.

Possible settings	ON	All print and work file data sets are automatically assigned to the batch access method AM=STD if the logical dataset name (defined by the DEST subparameter) is defined by job control (same behaviour as with Natural Version 2.2).
	OFF	Automatic print and work file assignment to AM=STD is done only if the file is not assigned to another access method, e.g. AM=NAF.
Default setting	OFF	
Dynamic specification	YES	
Specification within session	NO	

 Copyright © Software AG 2002

FC - Filler Character for INPUT Statement

This Natural profile parameter is for:

- mainframes
- UNIX/OpenVMS and Windows

It indicates the default filler character to be used for fields displayed by an INPUT statement.

Possible settings	any character	Default filler character. It is used to pre-fill fields with the attribute specification "AD=A" when fields are written to a terminal by an INPUT statement. For modifiable input fields (AD=M), it is used to fill the rest of the field.
Default setting	Mainframes: X’00’ (X’40’ for TTY or batch mode)	
	UNIX/OpenVMS and Windows: blank	
Dynamic specification	YES	
Specification within session	NO	

This setting also applies to modifiable input fields (AD=M).

FCDP - Filler Character for Dynamically Protected Input Fields

This Natural profile parameter is for:

- mainframes
- UNIX/OpenVMS and Windows

It allows you to suppress the display of filler characters for input fields that have been made write-protected dynamically (that is, to which the attribute AD=P has been assigned via a control variable). Depending on the setting of the FCDP profile parameter, dynamically protected input fields are displayed filled either with blanks or with the defined filler characters.

Possible settings	ON	Dynamically protected input fields are displayed filled with filler characters. This may suggest to the users that they could enter something in the fields.
	OFF	Dynamically protected input fields are displayed filled with blanks.
Default setting	ON	
Dynamic specification	YES	
Specification within session	YES	Within a Natural session, the profile parameter FCDP can be overridden by the **session parameter** FCDP.

Example:

```
DEFINE DATA LOCAL
1 #FIELD1 (A5)
1 #FIELD2 (A5)
1 #CVAR1  (C) INIT <(AD=P)>
1 #CVAR2  (C)
END-DEFINE
*
INPUT #FIELD1 (AD=Y'_' CV=#CVAR1)  /* field is protected
      #FIELD2 (AD=Y'_' CV=#CVAR2)  /* field is not protected
...
END
```

Execution of the above program will display the following:

FCDP=ON:

```
#FIELD1 _____ #FIELD2 _____
```

FCDP=OFF:

```
#FIELD1 #FIELD2 _____
```

 Copyright © Software AG 2002

FDIC - Predict System File

This Natural profile parameter is for:

- mainframes
- UNIX/OpenVMS and Windows

Mainframes

This parameter indicates the Adabas database ID, file number, password and cipher key for the Predict system file. This file is the Adabas file which the Predict system uses to retrieve and/or store data. It is also used to store Natural DDMs.

Possible settings	see below	Adabas database ID, file number, password and cipher key for the Predict system file.
Default setting	none	
Dynamic specification	YES	
Specification within session	NO	

The syntax for this parameter is:

`FDIC=(`*`DBID`*`,`*`file-number`*`,`*`password`*`,`*`cipher-key`*`,RO)`

The DBID must be in the range from 1 to 254, or 256 to 65535. The file number must be in the range from 1 to 65535. The password must be a setting of 1 to 8 characters. The cipher key must be a setting of 8 characters.

The password is only required if the Predict system file has been password-protected using the Adabas security feature. The cipher key is only required if the Predict system file has been ciphered using the Adabas security feature.

"RO" indicates that the Predict system file is "read-only" and is only specified if modifications on the file are to be disabled.

Examples:

```
FDIC=(10,5,PASSW1,12345678)
FDIC=(1,200,,12345678)
FDIC=(1,5)
FDIC=(,5)
```

If any component of the FDIC setting is not specified, the corresponding setting of the parameter DBID, FNR, SYSPSW or SYSCIP applies for the Predict system file.

If you specify the FDIC parameter dynamically in conjunction with any of the parameters DBID, FNR, SYSPSW and SYSCIP, you must specify the FDIC parameter **after** any of these other parameters.

FDIC with VSAM System Files

For Natural with VSAM system files, the password is used to specify the logical name (DD or DLBL) of the system file as defined to VSAM.

Example:

```
FDIC=(10,5,SYSVSAM)
```

For further information, see Using Natural with VSAM System Files (in the Natural for VSAM documentation).

UNIX/OpenVMS and Windows

This parameter indicates the Adabas database ID, file number, password, cipher key and read-only flag for the Predict system file. This file is the Adabas file which Predict uses to retrieve and/or store data.

Possible settings	see below	Adabas database ID, file number, password, cipher key and read-only flag for the Predict system file.
Default setting	none	
Dynamic specification	YES	
Specification within session	NO	

The syntax for this parameter is:

FDIC=(*DBID*,*file-number*,*password*,*cipher-key*)

The DBID must be in the range from 0 to 65535, except 255. The file number must be in the range from 1 to 5000, password and cipher key must be settings of 1 to 8 characters. If no FDIC or FSEC is available, do not enter anything in the DBID and file number fields.

The password is only required if the Predict system file has been password-protected using the Adabas security feature. The cipher key is only required if the Predict system file has been ciphered using the Adabas security feature.

Since the Adabas security feature expects the password to be in upper case, it must be specified in upper-case letters.

Examples:

```
FDIC=(10,5,PASSW1,12345678)
FDIC=(1,200,,12345678)
FDIC=(1,5)
```

For information on system files, refer to System File Simulation.

 Copyright © Software AG 2002

FNAT - Natural System File for System Programs

This Natural profile parameter is for:

- mainframes
- UNIX/OpenVMS and Windows

Mainframes

This parameter indicates the database ID, file number, password and cipher key for the Natural system file for Natural system programs. The Natural system file is the database file from which all Natural system programs are retrieved and upon which all system commands operate. Error texts and Natural help information are also contained in this system file.

Possible settings	Database ID: 1 to 254, or 256 to 65535 FNR: 1 to 65535 password: 1 to 8 characters cipher key: 8 characters	Database ID, file number, password and cipher key for the Natural system file for Natural system programs.
Default setting	none	
Dynamic specification	YES	
Specification within session	NO	

The syntax of this parameter is:

`FNAT=(`*`DBID`*`,`*`file-number`*`,`*`password`*`,`*`cipher-key`*`,RO)`

The password is only required if the Natural system file has been password-protected using the Adabas security feature. The cipher key is only required if the Natural system file has been ciphered using the Adabas security feature.

"RO" indicates that the Natural system file is "read-only" and is only specified if modifications on the file are to be disabled.

Examples:

```
FNAT=(,8)
FNAT=(22,5,PASSW2)
```

If any component of the FNAT setting is not specified, the corresponding setting of the parameter DBID, FNR, SYSPSW, SYSCIP or ROSY applies for the Natural system file for system programs.

If you specify the FNAT parameter dynamically in conjunction with any of the parameters DBID, FNR, SYSPSW, SYSCIP or ROSY, you must specify the FNAT parameter **after** any of these other parameters.

If you reorganize an Adabas FNAT file or if you unload/load data from the FNAT file (e.g. using ADAULD/ADALOD), you must specify USERISN=YES for the ADALOD utility.

FNAT with VSAM System Files

For Natural with VSAM system files, the password is used to specify the logical name (DD or DLBL) of the system file as defined to VSAM.

Example:

```
FNAT=(22,5,SYSVSAM)
```

For further information, see Using Natural with VSAM System Files (in the Natural for VSAM documentation).

UNIX/OpenVMS and Windows

This parameter indicates the database ID, file number, password, cipher key and read-only flag for the Natural system file for Natural system programs.

The Natural system file is the file from which all Natural system programs are retrieved and upon which all system commands operate. Error texts and Natural help information related to the Natural system libraries are also contained in this system file.

Possible settings	Database ID 1 to 254 FNR 1 to 255 password / cipher key 1 to 8 characters	Database ID, file number, password, cipher key and read-only flag for the Natural system file for Natural system programs. **Note:** The password and cipher key are reserved for future use; currently, they are ignored.
Default setting	none	
Dynamic specification	YES	
Specification within session	NO	

The syntax of this parameter is:

FNAT=(*DBID*,*file-number*,*password*,*cipher-key*)

Example:

```
FNAT=(22,5)
```

For information on system files, refer to System File Simulation.

 Copyright © Software AG 2002

FNR - Default File Number of Natural System Files

This Natural profile parameter is for mainframes only.

It identifies the default number of the file in which the Natural system files (FNAT, FUSER, FDIC, FSEC, FSPOOL) are located.

Possible settings	1 - 65535	File number. It applies to all Natural system files for which no individual file numbers are specified.
Default setting	none	
Dynamic specification	YES	If you specify the FNR parameter dynamically in conjunction with any of the individual system file parameters FNAT, FUSER, FDIC, FSEC and FSPOOL, you must specify the FNR parameter **before** any individual system file parameter.
Specification within session	YES	

File numbers for individual system files can be specified with the parameters FNAT, FUSER, FDIC, FSEC and FSPOOL.

Example 1:

```
FNR=5,DBID=10,FUSER=(,8)
```

This example assigns the user-program system file to File 8 on Database 10. All other system files are assigned to File 5 on Database 10.

Example 2:

```
FUSER=(,8),FNR=5,DBID=10
```

This example assigns all system files to File 5 on Database 10.

FS - Default Format/Length Setting for User-Defined Variables

This Natural profile parameter is for:

- mainframes
- UNIX/OpenVMS and Windows

It only applies to reporting mode; it has no effect in structured mode.

This parameter indicates whether a default format/length setting is to be in effect for the definition of user-defined variables in reporting mode.

Possible settings	ON	No default format/length is assigned by Natural. The format/length of all user-defined variables must be explicitly specified.
	OFF	A user-defined variable in a Natural program for which no format/ length is specified is assigned the default format/length "N7".
Default setting	ON	
Dynamic specification	YES	
Specification within session	YES	Within a Natural session, the profile parameter FS can be overridden by the **session parameter** FS.

 Copyright © Software AG 2002

FSEC - Natural Security System File

This Natural profile parameter is for:

- mainframes
- UNIX/OpenVMS and Windows

It only applies if Natural Security is used.

Mainframes

This parameter indicates the database ID, file number, password and cipher key for the Natural Security system file. This system file is the Adabas file from/in which all Natural Security information is retrieved/stored.

Possible settings	Database ID	1 to 254, or 256 to 65535	Database ID, file number, password and cipher key for the Natural Security system file
	FNR	1 to 65535	
	password	1 to 8 characters	
	cipher key	8 characters	
Default setting	none		
Dynamic specification	YES		
Specification within session	NO		

The syntax of this parameter is:

`FSEC=(`*`DBID`*`,`*`file-number`*`,`*`password`*`,`*`cipher-key`*`,RO)`

The password is only required if the Natural Security system file has been password-protected using the Adabas security feature. The cipher key is only required if the Natural Security system file has been ciphered using the Adabas security feature.

"RO" indicates that the Natural Security system file is "read-only" and is only specified if modifications on the file are to be disabled.

Example:
`FSEC=(10,8)`

If any component of the FSEC setting is not specified, the corresponding setting of the parameter DBID, FNR, SYSPSW, SYSCIP or ROSY applies for the Natural Security system file.

If you specify the FSEC parameter dynamically in conjunction with any of the parameters DBID, FNR, SYSPSW, SYSCIP or ROSY, you must specify the FSEC parameter **after** any of these other parameters.

UNIX/OpenVMS and Windows

Possible settings	Database ID	1 to 65535, except 255	Database ID, file number, password and cipher key for the Natural Security system file
	FNR	1 to 5000	
	password / cipher key	1 to 8 characters	
Default setting	none		
Dynamic specification	YES		
Specification within session	NO		

Note: For a non-security environment FSEC must be set inactive: **DBID = 0 and FNR = 0**

The syntax of this parameter is:

`FSEC=(`*`DBID`*`,`*`file-number`*`,`*`password`*`,`*`cipher-key`*`)`

The DBID must be in the range from 1 to 65535, except 255. The file number must be in the range from 1 to 5000, password and cipher key must be settings of 1 to 8 characters. If no FDIC or FSEC is available, do not enter anything in the DBID and file number fields.

The password is only required if the Natural Security system file has been password-protected using the Adabas security feature. The cipher key is only required if the Natural Security system file has been ciphered using the Adabas security feature.

Since the Adabas security feature expects the password to be in upper case, it must be specified in upper-case letters.

Examples:

FSEC=(10,8)
FSEC=10,5,PASSW1,12345678
FSEC=1,200,,12345678

For information on system files, refer to System File Simulation.

 Copyright © Software AG 2002

FSPOOL - Natural Advanced Facilities Spool File

This Natural profile parameter is for mainframes only.

It only applies to Natural Advanced Facilities. It indicates the database ID, file number, password and cipher key for the Natural Advanced Facilities spool file. The spool file is the database file that is used by Natural Advanced Facilities. This file must be different from the FNAT, FUSER, FDIC and FSEC system files.

Possible settings	Database ID	1 to 254, or 256 to 65535	Database ID, file number, password and cipher key for the Natural Advanced Facilities spool file.
	FNR	1 to 65535	The password is only required if the spool file has been password-protected using the Adabas security feature. The cipher key is only required if spool file has been ciphered using the Adabas security feature.
	password	1 to 8 upper-case characters	
	cipher key	8 characters	If any component of the FSPOOL setting is not specified, the corresponding setting of the parameter DBID, FNR, SYSPSW or SYSCIP applies for the spool file.
Default setting	none		
Dynamic specification	YES		If you specify the FSPOOL parameter dynamically in conjunction with any of the parameters DBID, FNR, SYSPSW or SYSCIP, you must specify the FSPOOL parameter **after** any of these other parameters.
Specification within session	NO		

The syntax for this parameter is:

`FSPOOL=(`*`dbid`*`,`*`file-number`*`,`*`password`*`,`*`cipher-key`*`,RO)`

"RO" indicates that the spool file is "read-only" and is only specified if modifications on the file are to be disabled. This would mean, for example, that no reports could be stored on the spool file.

Example:

```
FSPOOL=(10,8)
```

FSPOOL with VSAM System Files

For Natural with VSAM system files, the password is used to specify the logical name (DD or DLBL) of the system file as defined to VSAM.

Example:

```
FSPOOL=(10,8,SYSVSAM)
```

For further information, see Using Natural with VSAM System Files (in the Natural for VSAM documentation).

FUSER - Natural System File for User Programs

This Natural profile parameter is for:

- mainframes
- UNIX/OpenVMS and Windows

It indicates the database ID, file number, password and cipher key for the Natural user-program system file. This system file is the database file from which all user-written Natural programs are retrieved.

Mainframes

Possible settings	Database ID	1 to 254, or 256 to 65535	Database ID, file number, password and cipher key for the Natural user-program system file. The password is only required if the Natural user-program system file has been password-protected using the Adabas security feature. The cipher key is only required if the Natural user-program system file has been ciphered using the Adabas security feature. If any component of the FUSER setting is not specified, the corresponding setting of the parameter DBID, FNR, SYSPSW, SYSCIP or ROSY applies for the Natural user-program system file.
	FNR	1 to 65535	
	password	1 to 8 upper-case characters	
	cipher key	8 characters	
	none		
Default setting	ON		
Dynamic specification	YES		If you specify the FUSER parameter dynamically in conjunction with any of the parameters DBID, FNR, SYSPSW, SYSCIP or ROSY, you must specify the FUSER parameter **after** any of these other parameters.
Specification within session	NO		

The syntax of this parameter is:

`FUSER=(`*`DBID`*`,`*`file-number`*`,`*`password`*`,`*`cipher-key`*`,RO)`

"RO" indicates that the Natural user-program system file is "read-only" and is only specified if modifications on the file are to be disabled.

Examples:

```
FUSER=(,8)
FUSER=(22,5,PASSW2)
```

 Copyright © Software AG 2002

FUSER with VSAM System Files

For Natural with VSAM system files, the password is used to specify the logical name (DD or DLBL) of the system file as defined to VSAM.

Example: `FUSER=(22,5,SYSVSAM)`

For further information, see Using Natural with VSAM System Files (in the Natural for VSAM documentation).

UNIX/OpenVMS and Windows

Possible settings	Database ID	1 to 254	Database ID, file number, password, cipher key and read-only flag for the Natural user-program system file. **Note:** The password and cipher key are reserved for future use; currently, they are ignored.
	FNR	1 to 255	
	password /cipher key	1 to 8 upper-case characters	
	none		
Default setting	ON		
Dynamic specification	YES		If you specify the FUSER parameter dynamically in conjunction with any of the parameters DBID, FNR, SYSPSW, SYSCIP or ROSY, you must specify the FUSER parameter **after** any of these other parameters.
Specification within session	NO		

The syntax of this parameter is:

`FUSER=(`*`DBID`*`,`*`file-number`*`,`*`password`*`,`*`cipher-key`*`)`

Example:

`FUSER=(22,5)`

For information on system files, refer to System File Simulation.

GFID - Global Format IDs

This Natural profile parameter is for UNIX/OpenVMS and Windows platforms only.

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

Possible settings	ON	Global format IDs are generated for all views.
	OFF	Global format IDs are not generated.
	VID	Global format IDs are generated only for views in local/global data areas, but not for views defined within programs.
Default setting	ON	
Dynamic specification	YES	
Specification within session	NO	

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

 Copyright © Software AG 2002

HCAM - Hardcopy Access Method

This Natural profile parameter is for mainframes only.

It determines which access method is to be used for hardcopy output processing. You can specify one of the following access-method names:

Possible settings	Name:	Access method:
	STD	Batch, TIAM, TSO, CMS.
	COMP	Com-plete.
	CMS	CMS.
	IMS	Hardcopy is output directly to an IMS printer.
	NAF	Natural Advanced Facilities.
	USER	Third-party vendor print interface.
	ANY	Hardcopy output processing will be handled by the first access method available the search sequence for available access methods is the sequence in which the access methods are listed here).
	OFF	Hardcopy output processing will not be handled by any access method.
Default setting	ANY	
Dynamic specification	YES	
Specification within session	NO	

Note for BS2000/OSD Users:
HCAM=STD is a necessary setting for routing hardcopy output to standard print files.

The hardcopy output destination is specified by HCDEST.

HCDEST - Hardcopy Output Destination

This Natural profile parameter is for mainframes only.

It presets the hardcopy output destination for the terminal command %H (without the *destination* operand).

Possible settings	1 to 8 characters	Valid hardcopy output destination.
	blank	A default destination may in some environments be supplied by the operating system or TP monitor.
Default setting	blank	
Dynamic specification	YES	
Specification within session	YES	This hardcopy output destination can be overwritten during a session by specifying "%H*destination*"; see also the terminal command %H.

If you are running Natural under TSO or in batch mode, the CMHCOPY dataset must be defined in the JCL or by dynamic allocation. The same applies to any other hardcopy dataset specified by HCDEST.

Under TSO, the hardcopy dataset specified by HCDEST is closed after %H at the next terminal I/O. The default CMHCOPY dataset is closed not at terminal I/O but at session termination.

 Copyright © Software AG 2002

HI - Help Character

This Natural profile parameter is for:

- mainframes
- UNIX/OpenVMS and Windows

It defines the character which is to be used to invoke a field-specific helproutine or a map helproutine (if defined for a given map).

Possible settings	any special character	The character which is to be used to invoke a field-specific helproutine or a map helproutine. The character specified with the HI parameter must not be the same as the one specified with the CF parameter (control character for mainframe terminal commands); it should not be the same as the one specified with the DC parameter (decimal character), IA parameter (input assign character) or ID parameter (input delimiter character).
	blank	Numeric fields which have a helproutine assigned are internally translated to alphanumeric format so as to make it possible for the user to enter a question mark into the field to invoke the helproutine. To prevent this internal translation (that is, if you wish to make sure that alphabetical characters cannot be entered into a numeric field) you can set the HI parameter to blank. However, when HI=' ' is set, a help key must be defined; otherwise it is not possible to invoke a helproutine for any field.
Default setting	?	
Dynamic specification	YES	
Specification within session	NO	

IA - Input Assign Character

This Natural profile parameter is for:

- mainframes
- UNIX/OpenVMS and Windows

It defines the character to be used as the assignment character for the input parameter processing in INPUT statements, either in keyword/delimiter mode or when processing data from the Natural stack.

Possible settings	any special character	Assignment character for the input parameter processing in INPUT statements. The character specified with the IA parameter must not be the same as the character specified with the CF parameter (control character for mainframe terminal commands), DC parameter (decimal character) or ID parameter (input delimiter character); it should not be the same as the one specified with the HI parameter (help character).
Default setting	=	
Dynamic specification	YES	
Specification within session	YES	Within a Natural session, the profile parameter IA can be overridden by the **session parameter** IA.

 Copyright © Software AG 2002

ID - Input Delimiter Character

This Natural profile parameter is for:

- mainframes
- UNIX/OpenVMS and Windows

It defines the character to be used as a delimiter character for the INPUT statement.

Possible settings	any special character	The character specified with this parameter must not be the same as the one specified with the DC parameter (decimal character) or IA parameter (input assign character); it should not be the same as the one specified with the CF parameter (control character for mainframe terminal commands) or HI parameter (help character). The period (**.**) should not be used as input delimiter, because this might lead to situations in which a program termination period would be misinterpreted as input delimiter. An asterisk (*****) should not be used either.
	blank	No input delimiter character is defined.
Default setting	, (comma)	If the input delimiter character is to be a comma, it must be specified as ID=’,’ when using the dynamic parameter facility, because the character "," separates individual parameters.
Dynamic specification	YES	
Specification within session	YES	Within a Natural session, the profile parameter ID can be overridden by the **session parameter** ID.

IDSIZE - Size of Buffer for Natural/IDMS

This Natural profile parameter is for mainframes only.

It only applies if the Natural/IDMS Interface is installed. It determines the size of the Natural work buffer for the IDMS database system.

Possible settings	0 - 64	Buffer size in KB. If Natural for IDMS is to be used, this parameter must be specified; see the Natural for IDMS documentation for further information.
Default setting	0	Natural for IDMS is not used.
Dynamic specification	YES	
Specification within session	NO	

If IDSIZE=0 or if the requested space is not available, the Natural/IDMS Interface cannot be used.

 Copyright © Software AG 2002

IKEY - Processing of PA and PF Keys

This Natural profile parameter is for:

- mainframes
- UNIX/OpenVMS and Windows

It indicates the action to be taken when a video-terminal program-attention key (PA key) or program-function key (PF key) is used to enter data, and the key has not been defined to the Natural program with the SET KEY statement.

Mainframes

Possible settings	ON	The setting "ENTR" is placed in the Natural system variable *PF-KEY; that is, Natural reacts as if ENTER had been pressed.
	OFF	A REINPUT message is generated, prompting the user to press a valid key.
Default setting	OFF	
Dynamic specification	YES	
Specification within session	NO	

UNIX/OpenVMS and Windows

Possible settings	ON	Natural reacts as if ENTER had been pressed.
	OFF	A REINPUT message is generated, prompting you to press a valid key.
Default setting	OFF	
Dynamic specification	YES	
Specification within session	NO	

IM - Input Mode

This Natural profile parameter is for:

- mainframes
- UNIX/OpenVMS and Windows

It indicates the default mode for video-terminal input.

Possible settings	F	Forms mode.
	D	Delimiter mode.
Default setting	F	
Dynamic specification	YES	
Specification within session	YES	Within a Natural session, the setting of the profile parameter IM can be changed at any time with the Natural **terminal commands** " %D" and "%F".

 Copyright © Software AG 2002

IMSG - Session Initialization Error Messages

This Natural profile parameter is for mainframes only.

It is used to suppress the initialization error-messages screen. It can be useful to avoid undesired output, for example, for printer sessions. As error diagnosis may become difficult, use this parameter with caution.

Possible settings	ON	The initialization error messages screen is displayed in the case of an error.
	OFF	The initialization error messages screen is not displayed.
Default setting	ON	
Dynamic specification	YES	
Specification within session	NO	

INIT-LIB - Library for Automatic Logon

This Natural profile parameter is for UNIX/OpenVMS and Windows platforms only.

It specifies the name of the library to be used for an automatic logon (see the parameter AUTO) when Natural is started.

Possible settings	1-8 characters	Valid library name.
Default setting	none	
Dynamic specification	NO	
Specification within session	NO	

If Natural Security is installed, INIT-LIB is not evaluated; the library to be used for automatic logon is read from the FSEC system file (see the Natural Security documentation for further information).

 Copyright © Software AG 2002

INTENS - Printing of Intensified Fields

This Natural profile parameter is for mainframes only.

It indicates how many times an intensified field or the underline character is to be overprinted when it is printed on a print device.

Possible settings	1 - 10	Number of times an intensified field or the underline character is overprinted. The underline character is printed only if the parameter is set greater than 1. With INTENS=1, underlined fields are printed without underlining.
Default setting	3	
Dynamic specification	YES	
Specification within session	NO	

ISIZE - Size of Initialization Buffer

This Natural profile parameter is for mainframes only.

It determines the size of the Natural initialization buffer. This buffer is used to hold the parameters Natural is initialized with, as well as the work areas and tables used by Natural during the initialization.

Possible settings	4-32	Buffer size in KB.
Default setting	8	
Dynamic specification	NO	
Specification within session	NO	

The ISIZE parameter is ignored if it is specified in a parameter string activated by a SYS or PROFILE profile parameter or in an alternative parameter module (as specified with the PARM profile parameter).

 Copyright © Software AG 2002

ITERM - Session Termination in Case of Initialization Error

This Natural profile parameter is for mainframes only.

It determines whether or not the Natural session is to continue in the case of a session initialization error.

Possible settings	ON	If a session initialization error occurs, the session is terminated immediately after the initialization error messages.
	OFF	If session initialization errors occur, the following happens: In online mode: the initialization errors are displayed and you can choose to either continue or terminate the session. In batch mode: the session is continued with the initialization errors going unnoticed - possibly leading to errors or undesired results later in the session. The setting ITERM=OFF is not possible with STACK=INPL.
Default setting	ON	
Dynamic specification	YES	
Specification within session	NO	

ITRACE - Internal Trace Function

This Natural profile parameter is for mainframes only.

Do not use this parameter without prior consultation of Software AG Support.

It is used to activate/deactivate the internal trace function. The internal trace function is intended primarily for Software AG internal use for debugging purposes.

Possible settings	ON	Trace data is passed to the SYSRDC utility.
	OFF	No trace data is passed to the SYSRDC utility.
Default setting	ON	
Dynamic specification	YES	
Specification within session	YES	Within a Natural session, the terminal command "%TRI" can be used to activate/ deactivate the internal trace function.

 Copyright © Software AG 2002

KAPRI - Kanji Printing

This Natural profile parameter is for mainframes only.

It is used to specify parameters for Kanji printing. It corresponds to the NTKAPRI macro in the parameter module.

Possible settings	ON	Printing of Kanji output is supported.
	OFF	If you do not wish to use the parameter list of the NTKAPRI macro, you specify KAPRI=OFF.
Default setting	ON	
Dynamic specification	YES	This parameter can only be specified dynamically.
Specification within session	NO	

The KAPRI parameter is specified as follows:
KAPRI=’*parameter*,...’
The parameters specified with the KAPRI parameter will be appended to the list of parameters specified with the NTKAPRI macro.

With a subsequent KAPRI parameter, you can then specify a new parameter list.

For further information, please refer to the Kanji print interface documentation (available from Software AG of Far East).

KC - Check for Statement Keywords

This Natural profile parameter is for UNIX/OpenVMS and Windows platforms only.

It controls Natural programs during compilation for Natural statement keywords which are invalidly used as variable names.

Possible settings	ON	The check for statement keywords is performed.
	OFF	No check for statement keywords is performed.
Default setting	OFF	
Dynamic specification	YES	
Specification within session	NO	

The section Keywords and Reserved Words contains a list of all Natural keywords and reserved words, in which the statement keywords affected by the KC parameter are marked.

 Copyright © Software AG 2002

KEY - Setting Assignments to PA, PF and CLEAR Keys

This Natural profile parameter is for:

- mainframes
- UNIX/OpenVMS

It is used to assign settings to the CLEAR key, program attention keys (PA keys) and program function keys (PF keys) on video terminals.

Possible settings	any character string	Settings can be assigned to the keys PA1 to PA3, PF1 to PF24 and to the CLEAR key. The setting assigned to each key can be any character string. The character string must represent a Natural system command or a user command (user program). If the setting contains embedded blanks, it must be enclosed in apostrophes.
Default setting	none	
Dynamic specification	NO	
Specification within session	YES	

Assignments made with the KEY parameter are only valid from the Natural NEXT prompt (Mainframe)/ from within the Natural Direct Command window (UNIX/OpenVMS).

The entire string specified with the KEY parameter must be enclosed in parentheses (except KEY=OFF). KEY=OFF un-assigns all keys.

Examples for Mainframe:

```
KEY= (PF4=OFF,PF1=HELP,PF3='EDIT MAP',PF2=USERPGM1,CLR=LOGOFF)
```

```
KEY=OFF
```

```
KEY PF4=OFF
KEY PF3="EDIT MAP"
KEY CLR=LOGOFF
KEY OFF
```

LC - Lower to Upper Case Translation

This Natural profile parameter is for:

- mainframes
- UNIX/OpenVMS and Windows

This parameter controls lower-case to upper-case translation of input characters.

To disable or enable lower-case to upper-case translation dynamically within the active Natural session, you should use the terminal commands %L or %U, respectively.

Mainframes

Possible settings	ON	No translation of lower-case characters to upper case is performed.
	OFF	All lower-case characters entered from a video terminal are translated to upper case by Natural.
Default setting	OFF	
Dynamic specification	YES	
Specification within session	YES	

Lower/upper-case translation can also be performed by a TP monitor before control is given to Natural. The corresponding TP-monitor parameters for lower/upper-case translation also have to be reviewed to ensure correct translation.

A user-supplied translation table can be used to perform translation from lower case to upper case; see NTUTAB1 Macro.

UNIX/OpenVMS and Windows

Possible settings	ON	No translation of lower-case characters to upper case is performed.
	OFF	All lower-case characters entered from a video terminal are translated to upper case by Natural.
Default setting	ON	
Dynamic specification	YES	
Specification within session	NO	

 Copyright © Software AG 2002

LDB - Wait Time for Response of Local Adabas Database

This Natural profile parameter is for Windows platforms only.

It specifies the time limit Natural is to wait for a response of the local Adabas database.

Possible settings	0 - 999	Time limit in seconds. If it is exceeded, an appropriate error message is issued. If "0" is specified, no time limit will be in effect.
Default setting	30	
Dynamic specification	NO	
Specification within session	NO	

If Entire Net-work is installed, the specified time limit also affects Entire Net-work’s timeout processing.

LE - Reaction when Limit for Processing Loop Exceeded

This Natural profile parameter is for:

- mainframes
- UNIX/OpenVMS and Windows

It controls the action to be taken if the limit specified for processing loop execution is exceeded.

Possible settings	ON	Loop execution is aborted and an error message is issued at the end of the Natural program.
	OFF	Loop execution is aborted and processing continues without an error message.
Default setting	OFF	
Dynamic specification	YES	
Specification within session	YES	Within a Natural session, the profile parameter LE can be overridden by the **session parameter** LE.

 Copyright © Software AG 2002

LFILE - Dynamic Specification of Logical File

This Natural profile parameter is for:

- mainframes
- UNIX/OpenVMS and Windows

It dynamically specifies information concerning the physical file to be associated with any logical file.

Mainframes

Possible settings	database ID, FNR and, if applicable, password and cipher key	System file location.
Default setting	none	
Dynamic specification	YES	This parameter can only be specified dynamically.
Specification within session	NO	

This parameter can be used for Software AG products which have their own system files (for example, Con-nect, Natural Elite) to specify where such a system file is to be located. These products use database ID "255" and a logical file number (FNR) in their DDMs. With the LFILE parameter you specify which physical database ID (DBID) and file number (and, if applicable, password and cipher key) are associated with that logical file number.

LFILE can also be used to define a so-called scratch-pad file with logical file number "212"; see also the profile parameter ROSY and refer to Natural Scratch-Pad File.

The LFILE parameter is specified as follows:

`LFILE=(`*`logical-FNR`*`,`*`physical-DBID`*`,`*`physical-FNR`*`,`*`password`*`,`*`cipher-key`*`,RO)`

The logical file number must be in the range from 1 to 251. The DBID must be in the range from 0 to 254, or 256 to 65535. The physical file number must be in the range from 1 to 65535. The password must be a setting of 1 to 8 characters. The cipher key must be a setting of 8 characters.

"RO" indicates that the file is "read-only" and is only specified if modifications on the file are to be disabled.

Natural records the physical file information and uses it for any Adabas calls to:
DBID=255,FNR=*logical-FNR*

The LFILE parameter can be specified several times to specify different logical files.

The LFILE parameter corresponds to the NTFILE Macro, which can be specified in the Natural parameter module (NATPARM).

UNIX/OpenVMS and Windows

Possible settings	see text	
Default setting	none	
Dynamic specification	YES	
Specification within session	NO	

LFILE can be used for Software AG products which have their own system files to specify where such a system file is to be located. Such products use database ID "255" and a logical file number (FNR) in their DDMs. With the LFILE parameter you specify which physical database ID (DBID), file number (FNR), password and cipher key are associated with this logical file number.

Password and Cipher Key

Password and cipher key must be settings of 1 to 8 characters. They are only required if the file has been password-protected and/or ciphered using the Adabas security feature.

Since the Adabas security feature expects the password to be in uppercase, it must be specified in uppercase letters.

> **Note:**
> With FNAT and FUSER, the password and the cipher key are reserved for future use; currently they are ignored.

The LFILE parameter is specified as follows:

`LFILE=(`*`logical-ID`*`,`*`DBID`*`,`*`FNR`*`,`*`password`*`,`*`cipher-key`*`)`

The logical ID must in the range from 1 to 251, the physical database ID must be in the range from 0 to 65535 (except 255), the physical file number must be in the range from 1 to 5000, the password and cipher key must be settings of 1 to 8 characters.

Natural records the physical file information and uses it for any Adabas calls to:

`DBID=255,FNR=`*`logical-ID`*

The LFILE parameter can be specified several times so as to specify different logical files. Natural administrators can specify LFILE up to 255 times; other users can specify LFILE up to 8 times. A list of existing LFILE assignments is displayed in the "LFILE No" list box.

 Copyright © Software AG 2002

LFILMAX - Maximum Number of Dynamic Specifications of LFILE Parameter

This Natural profile parameter is for UNIX/OpenVMS and Windows platforms only.

It can be used to specify the maximum number of dynamic specifications of the profile parameter LFILE.

Possible settings	0 - 255	Maximum number of dynamic specifications of the profile parameter LFILE.
Default setting	8	
Dynamic specification	NO	
Specification within session	NO	

LIBNAM - Name of External Program Load Library

This Natural profile parameter is for mainframes and only applies under BS2000/OSD, OS/390 batch mode, and TSO.

It specifies the name of the load library from which programs are to be loaded dynamically when using Natural under BS2000/OSD, OS/390 batch mode, or TSO.

Possible settings	character string	Any valid BS2000/OSD file name, or 8-byte DDNAME of load library
Default setting	none	
Dynamic specification	YES	
Specification within session	NO	

Under OS/390, a JCL statement with a DDNAME that equals the LIBNAM setting also needs to be specified. By default, programs are loaded from the job steplib.

 Copyright © Software AG 2002

LOG (Internal Use)

This Natural profile parameter is for mainframes only.

This parameter is reserved for internal use by Natural. Do not change its setting.

LOGONRQ - Logon for RPC Server Request Required

This Natural profile parameter is for UNIX/OpenVMS and Windows platforms only.

Note for mainframes:
Use the keyword parameter LOGONRQ of the NTRPC macro for static specification, or the subparameter with the same name of the profile parameter RPC.

It determines whether or not a logon is required for an RPC server request. For further information, see Using Natural RPC with Natural Security in the Natural Remote Procedure Call documentation.

This parameter is specified on the server side only.

Possible settings	ON	A logon is required; that is, the server only accepts requests from clients which include logon data in the RPC server request. For conversational requests, the logon data is only necessary when the conversation is opened.
	OFF	A logon is not *required*. Logon data is nevertheless processed.
Default setting	ON	
Dynamic specification	YES	
Specification within session	NO	

For Natural clients the logon data can either be requested:

- by setting the LOGON option of the SYSRPC Service Directory Maintenance
- using the logon indicator of the profile parameter DFS.

For further information see the Natural Remote Procedure Call documentation.

 Copyright © Software AG 2002

LS - Line Size

This Natural profile parameter is for:

- mainframes
- UNIX/OpenVMS and Windows

It parameter indicates the maximum number of characters permitted per line for DISPLAY, INPUT , PRINT and WRITE statements. Together with the profile parameter PS, it defines the amount of page buffer space acquired during Natural initialization.

Possible settings	35 - 250, or 0	Maximum number of characters permitted per line. LS=0 indicates that the output device line size is to be used.
Default setting	0	In online mode, the default setting will always be set to the physical line size, even if LS has been set to another setting at the start of the Natural session.
Dynamic specification	YES	
Specification within session	NO	Within a Natural session, the profile parameter LS can be overridden by the **session parameter** LS.

Copyright © Software AG 2002

LSTEP - Initial Setting for *STEPLIB System Variable

This Natural profile parameter is for UNIX/OpenVMS and Windows platforms only.

It specifies the initial setting for the *STEPLIB system variable; see also System Variables.

Possible settings	1-8 characters	Any valid library name.
Default setting	STEP	
Dynamic specification	NO	
Specification within session	NO	

All existing steplibs are displayed. In addition, you can specify the name, DBID and FNR of the library where the Natural steplib is located. Up to eight steplibs can be defined.

 Copyright © Software AG 2002

LT - Limit for Processing Loops

This Natural profile parameter is for:

- mainframes
- UNIX/OpenVMS and Windows

It limits the number of Adabas records which can be read in processing loops within a Natural program.

Possible settings	Mainframes:	0 - 2147483647	Maximum number of records which can be read in any given processing loop. This limit applies to any processing loop initiated with a READ, FIND or HISTOGRAM statement. All records read (including rejected records from a WHERE clause) are counted against this limit.
	UNIX/OpenVMS and Windows:	0 - 99999999	
Default setting	99999999		
Dynamic specification	YES		
Specification within session	YES		Within a Natural session, the profile parameter LT can be overridden by the **session parameter** LT.

MADIO - Maximum DBMS Calls between Screen I/O Operations

This Natural profile parameter is for:

- mainframes
- UNIX/OpenVMS and Windows

It indicates the maximum number of DBMS calls permitted between two screen I/O operations (also in batch mode).

Possible settings	0, or 30 - 32767	Maximum number of DBMS calls. MADIO=0 indicates that no limit is to be in effect.
Default setting	512	
Dynamic specification	YES	
Specification within session	NO	

If the specified limit is exceeded, the Natural program is interrupted and the user is notified with an appropriate Natural error message (Mainframes: Natural Error Message 1009).

 Copyright © Software AG 2002

MAINPR - Override Default Output Report Number

This Natural profile parameter is for:

- mainframes
- UNIX/OpenVMS and Windows

Possible settings	0 - 31	Valid printer number.
Default setting	0	
Dynamic specification	YES	
Specification within session	NO	

Mainframes

With this parameter, you can separate program output from Natural system output, which may be useful particularly in batch mode.

This applies to program output for Report 0, as produced by DISPLAY, PRINT, WRITE or INPUT statements (except INPUT statements which contain input fields (AD=A or AD=M)).

If the MAINPR parameter is specified, program output for Report 0, which would normally be output on the printer assigned to Report 0, is output on the printer specified with MAINPR instead; while system output (NEXT prompt, DATA prompt, etc.) is always output on the primary output device (Report 0); the MAINPR setting must be a a valid printer number (0 - 31).

A logical printer corresponding to the report number specified must be defined to Natural. A printer is defined with the profile parameter PRINT, with the macro NTPRINT or automatically by JCL (in batch mode or under TSO).

The MAINPR parameter does not apply to output from system programs in the Natural system library "SYSLIB", which is always output on the primary output device (Report 0).

UNIX/OpenVMS and Windows

This parameter is used to override the default output report number for all Natural reports. It must be set to a valid printer number (0 - 31).

Specifying a MAINPR setting is the same as changing all of the DISPLAY, PRINT, WRITE or INPUT statements statement printer references from the default setting (0) to the desired printer number.

A physical printer corresponding to the report number specified must be defined to Natural (as described in the section Devices).

MAXBUFF - Maximum Buffer Size

This Natural profile parameter is for UNIX/OpenVMS and Windows platforms only.

Note for mainframes:
Use the keyword parameter MAXBUFF of the NTRPC macro for static specification, or the subparameter with the same name of the profile parameter RPC.

MAXBUFF can be specified on both the client and the server side.

On the server side it determines the size of the buffer provided by the server to receive the client request including data and to send back the result.
On the client side, it determines the size of the buffer provided for the automatic execution of Natural RPC calls. This buffer is used to build the client request including data and to receive the result. For further information, see Stubs and Automatic RPC Execution in the Natural Remote Procedure Call documentation.

Possible settings	0-16380, but smaller than or equal to RPCSIZE-4	Maximum buffer size in KB. Note that the transport layer you are using may have specific restrictions.
Default setting	0	No buffer is allocated.
Dynamic specification	YES	
Specification within session	NO	

Note:
If you are using EntireX Broker, the value of MAXBUFF must not be greater than the EntireX Broker attribute MAX-MSG. Additionally, if EntireX Broker uses Entire Net-work, MAXBUFF is restricted by the startup parameter IUBL and must not exceed 32 KB. However, a maximum of 30 KB is recommended.

For further information see the Natural Remote Procedure Call documentation.

 Copyright © Software AG 2002

MAXCL - Maximum Number of Program Calls

This Natural profile parameter is for:

- mainframes
- UNIX/OpenVMS and Windows

It determines the maximum number of program calls permitted between two screen I/O operations. If the specified limit is exceeded, the Natural program is interrupted and and the user is notified with an appropriate Natural error message (Mainframe: NAT1029).

Possible settings	0, or 10 - 32767	Maximum number of program calls. MAXCL=0 indicates that no limit is to be in effect.
Default setting	50	
Dynamic specification	YES	
Specification within session	NO	

MAXROLL - Number of CMROLL Calls Before Roll-Out

This Natural profile parameter is for mainframes and applies under Com-plete and CICS only.

It determines the number of CMROLL calls after which a roll-out of the Natural thread is to be performed.

Possible settings	0 - 32767	Number of CMROLL calls. MAXROLL=0 indicates that no conditional CMROLL requests are issued.
Default setting	128	
Dynamic specification	YES	
Specification within session	NO	

The MAXROLL parameter can be used to control the frequency of conditional CMROLL requests. For example, MAXROLL=128 means that a conditional CMROLL request is issued after every 128th statement at compilation.

In certain cases, the Natural nucleus issues a conditional CMROLL request (wait time = 0), particularly at compilation after each statement. This is done to reset the CPU time window (under Com-plete) in order to avoid an automatic cancel due to the CPU time limit being exceeded; however, this has a negative impact on performance.

 Copyright © Software AG 2002

MAXUSER - Maximum Number of Users

This Natural profile parameter is for UNIX/OpenVMS and Windows platforms only.

It determines the maximum number of users that can have simultaneous access to the buffer pool. This number determines the sizes of some internal tables stored inside the shared memory during startup.

Possible settings	1 - 5000	Maximum number of users.
Default setting	20	
Dynamic specification	NO	
Specification within session	NO	

MENU - Menu Mode

This Natural profile parameter is for mainframes only.

It is used to switch Natural menu mode on or off.

Possible settings	ON	Menu mode is switched on.
	OFF	Disables menu mode.
Default setting	ON	
Dynamic specification	YES	
Specification within session	YES	Within a Natural session, the MENU parameter can be overridden by the **system command** MAINMENU .

 Copyright © Software AG 2002

ML - Position of Message Line

This Natural profile parameter is for all platforms. In Natural for Windows, UNIX and OpenVMS, this profile parameter does not exist in the Configuration Utility. See session parameter ML.

It indicates the line to be used for the display applications which do not set the message line position explicitly by using the SET CONTROL ’M’ statement. For information on the operand ’M’, see also %M - Control of Message Line.

Possible settings	B	Natural messages are displayed at the bottom of the screen.
	T	Natural messages are displayed at the top of the screen.
Default setting	T (mainframe) B (Windows, UNIX, OpenVMS)	
Dynamic specification	YES	
Specification within session	YES	

MONSIZE - Size of SYSTP Monitor Buffer

This Natural profile parameter is for mainframes only.

It determines the size of the buffer used by the Monitor function of the SYSTP utility.

Possible settings	0-256	Buffer size in KB.
Default setting	0	If MONSIZE=0 or if the requested space is not available, the Monitor function of the SYSTP utility cannot be used.
Dynamic specification	YES	
Specification within session	NO	

 Copyright © Software AG 2002

MP - Maximum Number of Pages of a Report

This Natural profile parameter is for mainframes only.

It determines the maximum number of pages to be produced for a report.

Possible settings	0-32767	The setting specified is the number of physical pages and has no effect on the starting page number used. The program will be terminated with an error message if the MP setting is exceeded.
Default setting	0	no page limit is defined.
Dynamic specification	YES	
Specification within session	NO	

Within a Natural session, the setting of profile parameter MP can be reduced, but not increased by the FORMAT statement.

MSGSF - Display System Error Messages in Short/Full Format

This Natural profile parameter is for:

- mainframes
- UNIX/OpenVMS and Windows

It can be used to avoid truncation of Natural system error messages.

Possible settings	ON	System error messages will be displayed in full; that is, program name, line number and actual message text.
	OFF	System error messages will be displayed in short form; that is, only the actual message text will be displayed (but not the program name and line number).
Default setting	ON	
Dynamic specification	YES	
Specification within session	YES	Within a Natural session, the profile parameter MSGSF can be overridden by the **terminal command** %MSGSF.

By default, a Natural system error message consists of the following:

- the name of the program,
- the number of the line that caused the error,
- the actual text of the message.

Depending on the size of the window in which the message is displayed, the text may be truncated. With this parameter, you can avoid such truncation.

 Copyright © Software AG 2002

MT - Maximum CPU Time

This Natural profile parameter is for mainframes and only applies to programs executed in batch mode or under TSO.

It determines the maximum amount of CPU time which can be used by a Natural program.

Possible settings	0 - 9999999	Maximum amount of CPU time in seconds. MT=0 indicates that no Natural CPU time limit is in effect.
Default setting	60	
Dynamic specification	YES	
Specification within session	YES	Within a Natural session, the profile parameter MT can be overridden by the **session parameter** MT.

The limit for programs operating in interactive mode is controlled by the TP monitor in use.

The maximum setting that can be used is determined by the operating system environment. Any setting in excess of the maximum is reduced to the maximum supported by the operating system.

In system environments which do not support CPU time measurement, the limit is interpreted as elapsed time. The CPU time limit is ignored for systems without timer support.

If Natural Security is installed, the profile parameter MT can be overridden with Natural Security. With Natural Security, the maximum value for the profile parameter MT is 32767. To use a higher value as specified with the MT profile or session parameter, specify MT=0 within Natural Security.

Copyright © Software AG 2002

NAFSIZE - Size of Buffer for Natural Advanced Facilities

This Natural profile parameter is for mainframes and only applies if Natural Advanced Facilities is installed.

It specifies the size of the work buffer used by Natural Advanced Facilities.

Possible settings	0 - 64	Buffer size in KB.
Default setting	0	Disables Natural Advanced Facilities.
Dynamic specification	YES	
Specification within session	NO	

If Natural Advanced Facilities is to be used, a setting has to be specified for this parameter; see NATSPOOL Initialization (in the Natural Advanced Facilities documentation).

If the requested space is not available, Natural Advanced Facilities cannot be used.

 Copyright © Software AG 2002

NAFUPF - Natural Advanced Facilities User Profile

This Natural profile parameter is for mainframes and only applies if Natural Advanced Facilities is installed.

It is used to specify the user-profile name for Natural Advanced Facilities.

Possible settings	1 to 8 characters	Name of the user profile.
Default setting	none	
Dynamic specification	YES	
Specification within session	NO	

See NATSPOOL Initialization (in the Natural Advanced Facilities documentation).

NATLOG - Natural Log File

This Natural profile parameter is for UNIX/OpenVMS and Windows only.

It is used to log messages that will not (or could not) be written to the standard output in interactive mode or to the output file CMPRINT in batch mode.

Possible settings	OFF	Disables the log mechanism.
	ERR	Logs error messages.
	INF	Logs information and success messages.
	WRN	Logs warning messages.
	ALL	Logs all kind of messages.
Default setting	OFF	
Dynamic specification	YES	
Specification within session	NO	

The location of NATLOG file is the TEMP directory of Natural (specified in the local configuration file NATURAL.INI). If this path is unknown, Natural creates the file in the current directory.

Natural tries to create the following file name, if the user-id/et-id information could be retrieved:
NATURAL_<user-ID>_<et-ID>.LOG

If the user ID and et ID could not be retrieved, then the following file name is used: **NATURAL.LOG**

Example File Names:

A Natural batch process is running with USER-ID=SYSTEM and ETID=14 then the resultant file name is:
NATURAL_SYSTEM_14.LOG

If the USER-ID could not be retrieved (which is the case if an error occurs during the initialization phase of Natural), then the resultant file name is:
NATURAL.LOG

Platform-specific Examples

The below examples show the contents of a Natural log file. At top of the file, there is a header with some environment information, like Natural Version, used parameter file and so on. Now two entries follow. The first one is an informational entry which displays the I/O channels needed for real batch mode. The second entry shows an error message. Both messages are counted in the statistics summary.

- Example NATLOG output contents for OpenVMS
- Example NATLOG output contents for UNIX
- Example NATLOG output contents for Windows

Example NATLOG output contents for OpenVMS:

 Copyright © Software AG 2002

```
# ##################################################
#          N a t u r a l   L o g   F i l e
# ##################################################
#
# Logging started at : 24-Jul-2001 15:05:31.044
#                    :
# Natural Version    : V 4.1.2 DEVELOP SAG 2001
# Server Type        : (none)
# Device             : BATCH  (real)
# Parameter File     : NATPARM
#                    :
# User ID            : NATURAL
# ET ID              : TEST
# Network User ID    : NATURAL
#                    :
# Host Name          : ALF2
# Machine Class      : VMS
# Operating System   : OVMS/AXPV7.2
#                    :
# Process ID         : 22826C96
#                    :
# NATLOG Option      : ALL
#
# ##################################################
#
# ------------------------------------------------------------------------------
# 15:05:31.051  NATURAL     INFORMATIONAL  STATISTICS:  INF=1  WRN=0  ERR=0
# ------------------------------------------------------------------------------
setting of parameter CMSYNIN (command file)
ALF2$USER:[HOME.TMP]BATCH.CMD
setting of parameter CMOBJIN (input file)

setting of parameter CMPRINT (output file)
ALF2$USER:[HOME.TMP]BATCH.OUT
# ------------------------------------------------------------------------------
# 15:05:31.090  NATURAL     ERROR          STATISTICS:  INF=1  WRN=0  ERR=1
# ------------------------------------------------------------------------------
 NATURAL Startup Error:  42
 Batch mode driver error.
 Parameter CMOBJIN not set.
```

Example NATLOG output contents for UNIX:

```
# ##################################################
#          N a t u r a l   L o g   F i l e
# ##################################################
#
# Logging started at : 24-Jul-2001 15:08:15.023
#                    :
# Natural Version    : V 4.1.2 DEVELOP SAG 2001
# Server Type        : (none)
# Device             : BATCH  (real)
# Parameter File     : NATPARM
#                    :
# User ID            : NATURAL
# ET ID              : TEST
# Network User ID    : NATURAL
#                    :
# Host Name          : hpn2
# Machine Class      : UNIX
# Operating System   : HP_HPUX  B.10.20A
#                    :
# Process ID         : 17921
#                    :
# NATLOG Option      : ALL
#
# ##################################################
#
# ------------------------------------------------------------------------------
# 15:08:15.028  NATURAL     INFORMATIONAL  STATISTICS:  INF=1  WRN=0  ERR=0
# ------------------------------------------------------------------------------
setting of parameter CMSYNIN (command file)
$HOME/tmp/batch.cmd
setting of parameter CMOBJIN (input file)

setting of parameter CMPRINT (output file)
$HOME/tmp/batch.out
# ------------------------------------------------------------------------------
# 15:08:15.033  NATURAL     ERROR          STATISTICS:  INF=1  WRN=0  ERR=1
# ------------------------------------------------------------------------------
 NATURAL Startup Error:  42
 Batch mode driver error.
 Parameter CMOBJIN not set.
```

Example NATLOG output contents for Windows

 Copyright © Software AG 2002

```
# ####################################################
#           N a t u r a l   L o g   F i l e
# ####################################################
#
# Logging started at : 24-Jul-2001 15:25:33.058
#                    :
# Natural Version    : V 4.1.2 PL1 SAG 2001
# Server Type        : (none)
# Device             : BATCH  (real)
# Parameter File     : NATPARM
#                    :
# User ID            : NATURAL
# ET ID              : TEST
# Network User ID    : MYDOMAIN\NATURAL
#                    :
# Host Name          : PCNAT
# Machine Class      : PC
# Operating System   : WNT-X86 4.0 (1381)
#                    :
# Process ID         : 274
#                    :
# NATLOG Option      : ALL
#
# ####################################################
#
# ------------------------------------------------------------------------------
# 15:25:33.059  NATURAL     INFORMATIONAL  STATISTICS:  INF=1  WRN=0  ERR=0
# ------------------------------------------------------------------------------
setting of parameter CMSYNIN (command file)
D:\TEMP\syn37437.tmp
setting of parameter CMOBJIN (input file)

setting of parameter CMPRINT (output file)
D:\TEMP\out37437.tmp
# ------------------------------------------------------------------------------
# 15:25:33.060  NATURAL     ERROR          STATISTICS:  INF=1  WRN=0  ERR=1
# ------------------------------------------------------------------------------
 NATURAL Startup Error:  42
 Batch mode driver error.
 Parameter CMOBJIN not set.
```

NATVERS - Switching between Natural Environments

This Natural profile parameter is for UNIX/OpenVMS and Windows platforms only.

It enables you to access various Natural environments.*** und dies?:*** that is, various err, txt, bin, etc, samples and tmp subdirectories. ?*** The NATVERS value you enter is used by Natural to determine the SAG.INI section you want to use with your Natural session. By default the Natural installation creates a section in the SAG.INI file that points to the installed Natural environment.

The Natural installation automatically identifies the Natural version and creates the section [NATURAL=5.1.1] in your SAG.INI file that points to the installed Natural environment. This might look as in the following example:

```
[NATURAL-5.1.1]
PROD_API=C:\Program Files\Software AG\5.1.1\Bin\natprd32.dll
NATGUI_BMP=C:\Program Files\Software AG\Natapps\Pic
NATDIR=C:\Program Files\Software AG
NATINI=C:\Program Files\Software AG\5.1.1\Etc
[NATURAL-5.1.1-END]
```

If you want to access envrionments other than the one that was installed automatically, create a new section for each environment as shown in the following example:

```
[NATURAL-Server5.1.1]
NATVERS=5.1.1
PROD_API=\\Server\Program Files\Software AG\5.1.1\Bin\natprd32.dll
NATGUI_BMP=\\Server\Program Files\Software AG\Natapps\Pic
NATDIR=\\Server\Program Files\Software AG
NATINI=\\Server\Program Files\Software AG\5.1.1\Etc
[NATURAL-Server5.1.1-END]
```

Possible settings	any character string	Determines the Natural version.
Default setting	5.1.1	**Note:** A DEFAULT-VERSION entry in SAG.INI is no longer necessary.
Dynamic specification	YES	This parameter can only be specified dynamically.
Specification within session	NO	

Replaced settings are still contained in the SAG.INI file and can be used by specifying them with NATVERS to switch to the corresponding Natural environments.

 Copyright © Software AG 2002

NC - Use of Natural System Commands

This Natural profile parameter is for:

- mainframes
- UNIX/OpenVMS and Windows

It controls whether Natural system commands can be used during the Natural session or not.

Possible settings	ON	System commands cannot be used - except FIN, LAST, LOGOFF, LOGON, MAINMENU , RENUMBER, RETURN, SETUP and TECH.
	OFF	All system commands can be used.
Default setting	OFF	
Dynamic specification	YES (Mainframe) NO (Open Systems)	
Specification within session	YES	Within a Natural session, the profile parameter NC can be overridden by the **session parameter** NC.

Natural terminal commands and user-created commands (object module names) are not affected by the NC parameter.

If you have Natural Security installed, any system command restrictions you set with Natural Security are valid, regardless of the setting of the NC profile parameter.

NENTRY - Left/Right Alignment of Numeric Field Entries

This Natural profile parameter is for UNIX/OpenVMS only.

It determines where the entry in a numeric field is written: to the left or to the right.

Possible settings	LEFT	An entry in a numeric field is written to the left.
	RIGHT	An entry in a numeric field is written to the right.
Default setting	LEFT	
Dynamic specification	YES	
Specification within session	YES	

 Copyright © Software AG 2002

NISN (Internal Use)

This Natural profile parameter is for mainframes only.

This parameter is reserved for internal use by Natural. Do not change its setting.

NOAPPLERR - Suppress Message Number Prefix NAT

This Natural profile parameter is for UNIX/OpenVMS and Windows platforms only.

It is used to suppress the message number prefix "NAT" with user-supplied error messages.

Possible settings	none	If you enter NOAPPLERR in the command line, the prefix NAT is not displayed in error messages.
Default setting	none	
Dynamic specification	YES	This parameter can only be specified dynamically.
Specification within session	NO	

 Copyright © Software AG 2002

NOPROX - Specify Local Domains for Direct Addressing

This parameter is for UNIX/OpenVMS only.

This Natural profile parameter specifies the domain(s) which shall be addressed directly, that is, not via the proxy.

Possible settings	See example below.
Default setting	None
Dynamic specification	NO
Specification within session	NO

Example:

```
"*.software-ag.de; sagus.software-ag.com"
```

NUCNAME - Name of Shared Nucleus

This Natural profile parameter is for mainframes only.

With this parameter, you specify the name of the (environment-independent) shared nucleus if it is to be loaded dynamically and not linked to the environment-dependent nucleus.

Possible settings	1 to 8 characters	Valid load module name.
Default setting	none	
Dynamic specification	YES	By specifying this parameter dynamically, you are able to use different shared nuclei (for example, for production and for testing) together with the same environment-dependent nucleus without having to relink the nucleus.
Specification within session	NO	

The NUCNAME parameter is ignored if it is specified in a parameter string activated by a SYS or PROFILE profile parameter or in an alternative parameter module (as specified with the PARM profile parameter).

 Copyright © Software AG 2002

OBJIN - Use of CMOBJIN as Natural Input File

This Natural profile parameter is for mainframes and applies to batch mode only.

It indicates whether the CMOBJIN file is to be used for input data provided with the INPUT statement in batch mode.

Possible settings	Y	Data for a Natural INPUT statement are read from the CMOBJIN file.
	N	The CMOBJIN file is not used and any data for an INPUT statement are read from the CMSYNIN file.
	R	Natural determines which option has been selected for a particular session by the presence or absence of the CMOBJIN DD/FILE statement in the Natural execution JCL/JCS.
Default setting	R	
Dynamic specification	YES	
Specification within session	YES	

OPF - Overwriting of Protected Fields by Helproutines

This Natural profile parameter is for:

- mainframes
- UNIX/OpenVMS and Windows

With this parameter, you specify whether the content of a write-protected field (attribute AD=P) can be overwritten by a helproutine assigned to the field.

Possible settings	ON	A helproutine assigned to a field can overwrite the field's content, even if the field is write-protected.
	OFF	Helproutines cannot overwrite the contents of write-protected fields.
Default setting	ON	
Dynamic specification	YES	
Specification within session	YES	Within a Natural session, the profile parameter OPF can be overridden by the **session parameter** OPF.

The OPF parameter only applies to the field for which a helproutine is invoked; it does not affect parameters explicitly passed to the helproutine. This means that the OPF parameter takes no effect if the field for which help is invoked is also explicitly specified as a parameter to be passed to the helproutine.

In addition, in reporting mode you can change the OPF setting using the statement SET GLOBALS.

 Copyright © Software AG 2002

OPRB - Database Open/Close Processing

This Natural profile parameter is for:

- mainframes
- UNIX/OpenVMS and Windows

Mainframes

This parameter only applies to Adabas and VSAM databases.

It controls the use of database Open/Close commands during a Natural session.

Possible settings	OPRB=(*string*)	With this syntax, you specify an Open request for *all* databases.
	OPRB=(DBID=*nn1*,*string*,DBID=*nn2*,*string*, ...)	With this syntax, you specify an Open request for specific individual databases. As defined in the macro NTDB, the specified DBID identifies the type of database.
	OPRB=(*string*,DBID=*nn1*,*string*,DBID=*nn2*,*string*, ...)	With this syntax, you specify an Open request for specific individual databases and also a default Open request - the initial *string* - which applies to all databases for which you do not specify an individual *string*.
Default setting	none	
Dynamic specification	NO	
Specification within session	NO	

Generally, the OPRB parameter uses one of the above syntaxes (the possible contents of the *strings* depend on the database system).

Instead of using the OPRB parameter, you can also use the macro NTOPRB in the Natural parameter module NATPARM.

Dynamic OPRB with Natural Security

A dynamically specified OPRB parameter applies for all logons to libraries in whose security profiles no OPRB parameter is specified. For a logon to a library in whose security profile the OPRB parameter is specified, any dynamically specified OPRB parameter is ignored and the one from the security profile applies.

OPRB for VSAM

The *strings* which can be specified for VSAM databases are described under the OPRB Parameter for VSAM Databases in the Natural for VSAM documentation.

OPRB for Adabas

For Adabas databases, the OPRB parameter is required if either of the following conditions is true for the Natural session:

- An explicit list of Adabas files to be accessed/updated is to be provided. This is necessary, for example, if Adabas cluster updating or exclusive file control is to be requested.
- A single logical transaction is to span two or more Natural programs and, therefore, it is not desired to have Natural issue an END TRANSACTION and CLOSE command at the termination of any given Natural program.

If the OPRB parameter is omitted in the NATPARM module or OPRB=OFF is specified as a dynamic parameter, a Natural session commences with an Adabas Open command requesting UPD (access/update) to the Natural system file. Natural also issues RELEASE CID (Adabas RC) commands to release all ISN lists (ISN lists specified in a RETAIN clause of a Natural FIND statement are not released).

The Adabas record buffer to be used with the initial Adabas OP command can be explicitly provided. The format is similar to that used in an Adabas record buffer for the OP command with the exception that no blanks can be embedded, and the complete setting must be enclosed in parentheses (not apostrophes).

Example 1: `OPRB=(ACC=2,4,6,UPD=8.)` This specifies that Adabas Files 2,4 and 6 are to be made available for access only and that Adabas File 8 is to be made available for update (which also implies access).
Example 2: `OPRB=(EXU=1,2,3.)` This specifies that Adabas Files 1,2 and 3 are to be placed under exclusive control for this Natural session.

Combinations of the keywords ACC, UPD and EXU must follow the rules as defined in the relevant Adabas documentation. When these keywords are coded, Natural issues an OP command at the start of a Natural session and a CL at the end of the Natural session. At the end of a Natural program, only the required RC commands are issued to release held ISN lists.

In all of the above situations, the OP command, which is always issued at the start of a Natural session, contains in the Additions 1 field of the Adabas control block the User ID for the Natural session. In batch mode, this is the job name. In TP mode, this is the setting supplied at system initialization by the Natural interface module. In both cases, the setting used is available in the Natural system variable *INIT-USER.

UNIX/OpenVMS and Windows

This Natural profile parameter controls the use of the Adabas C "Open/Close" commands during a Natural session.

 Copyright © Software AG 2002

Possible settings	OFF	If the OPRB parameter is set to "OFF", a Natural session starts with an Adabas OP command requesting UPD (access/update) to the Natural system file. In addition, with each Natural program termination, Natural issues an END TRANSACTION statement if the Natural session is not at ET status. Natural also issues RELEASE CID (Adabas RC) commands to release all ISN lists (ISN lists specified in a RETAIN clause of a Natural FIND statement are not released).
	NOOPEN	If OPRB is set to "NOOPEN", Natural also issues an Adabas OP command requesting UPD (access/update) to the Natural system file at the start of a Natural session, but no END TRANSACTION statement is issued at the end of a Natural program. Commands are issued, however, to release all ISN lists (ISN lists specified in a RETAIN clause of a Natural FIND statement are not released). The NOOPEN setting thus enables a single logical transaction to span more than one Natural program.
	OPRB=(*string*)	You can specify an Open request in accordance with the syntax described below. See also the examples listed.
Default setting	OFF	
Dynamic specification	NO	
Specification within session	NO	

It is required if any of the following conditions are true:

- An explicit list of Adabas files to be accessed/updated is to be provided. This is necessary, for example, if Adabas cluster updating or exclusive file control is to be requested.
- A single logical transaction is to span two or more Natural programs and therefore, it is not desired that Natural issue an END TRANSACTION and CLOSE DIALOG statement at the termination of any given Natural program.
- The Adabas record buffer to be used with the initial Adabas OP command can be explicitly provided. To access databases you have to specify the DBIDs and file numbers with their corresponding access rights at the OPRB string.

Entries may not contain blanks, must be enclosed in parenthesis and must follow the rules defined in the relevant Adabas documentation.

In addition to the Adabas syntax, internal file numbers can be specified by using the "x-y" notation (that is, all numbers between x and y).

OPRB String Syntax

DBID=(*x*) Specifies the database for the following access right entries:

UPD=(*file-list*) Specifies update permission (read/write) for the files in the file list
ACC=*(file-list*) Specifies access permission (read) for the files in the file list
EXU=(*file-list*) Specifies exclusive update permission (exclusive read/write) for the files in the file list

The trailing record buffer dot ("**.**") can be omitted in the OPRB string because it is appended automatically.

DBID=0 specifies the default record buffer entry and can be omitted if it is the first DBID listed in the OPRB string. This default record buffer is taken if there is no specific entry for the requested database.

Combinations of the keywords ACC, UPD and EXU must follow the rules as defined in the relevant Adabas documentation. When these keywords are coded instead of "NOOPEN", Natural issues an OP command at the start of a Natural session and a CL command at the end of the session. At the end of a Natural program, only the required RC commands are issued to release held ISN lists.

Example 1:

(ACC=2,3,4,DBID=15,UPD=3,4,ACC=5)

The following entries were defined:

’UPD=3,4,ACC=5.’ for DB 15
’ACC=2,3,4.’ for other databases (DB 0)

Example 2:

(DBID=15,ACC=2-7)

The following entry was defined:

’ACC=2,3,4,5,6,7.’ for DB 15; access to other databases is permitted.

Example 3:

(DBID=0,ACC=2,3,4,5.)

The following entry was defined:

’ACC=2,3,4,5.’ for all databases (DB 0).

> **Note:**
> If you have Natural Security installed, open/close processing works the same way as without Natural Security; the OPRB parameter in the security profile is provided for future use only.

 Copyright © Software AG 2002

OPT - Control of Natural Optimizer Compiler

This Natural profile parameter is for mainframes only.

It only applies if the Natural Optimizer Compiler is to be used.

This parameter corresponds to the macro NTOPT in the parameter module. It can be used to specify dynamically the same options which you can specify statically with the macro NTOPT.

Possible settings		See the Dynamic Profile Parameter OPT in the Natural Optimizer Compiler documentation.
Default setting	OFF	
Dynamic specification	YES	This parameter can only be specified dynamically.
Specification within session	YES	

OUTDEST - Output Destination for Asynchronous Processing

This Natural profile parameter is for mainframes and only applies to Natural under CICS, Com-plete and UTM.

It specifies the destination to which any Natural error message produced by an asynchronous application is to be sent.

Possible settings	1 to 8 characters	Destination to which a Natural error message is sent.
Default setting	setting of profile parameter SENDER	
Dynamic specification	YES	
Specification within session	NO	

After an error message has been sent, Natural terminates the asynchronous session.

Under UTM, this parameter is used to specify the ID of the terminal where output from an asynchronous application is to be displayed.

When and how error messages/output from an asynchronous application are output depends on the respective TP monitor.

 Copyright © Software AG 2002

PARM - Alternative Parameter Module

This Natural profile parameter is for:

- mainframes
- UNIX/OpenVMS and Windows

Mainframes

This parameter specifies an object module containing profile parameter definitions.

Possible settings	1-8 characters	Module name.
Default setting	none	
Dynamic specification	YES	
Specification within session	NO	

These definitions are coded using the various macros as described under Assembling a Natural Parameter Module. The macros are then assembled, resulting in an object module whose name is specified by the user.

When the PARM parameter is specified (either in the linked parameter module or as a dynamic parameter at Natural session start), the appropriate object module is loaded and the profile parameter definitions contained therein take effect. The parameter module is loaded dynamically from the steplib. Under CICS, a PPT entry is required for this parameter module.

Any profile parameter definitions in effect before the PARM parameter is processed (for example, definitions contained in the linked parameter module or prior dynamic parameters) are overridden when the specified parameter module is loaded.

To restrict the use of an alternative parameter module, you can use the macro NTUSER.

UNIX/OpenVMS and Windows

This parameter specifies a parameter file that is to be used instead of the default parameter file NATPARM and Read Access to NATPARM.SAG in the subdirectory PROF31 is required.

Possible settings	1-8 characters	Any valid file name.
Default setting	none	
Dynamic specification	YES	This parameter can only be specified dynamically.
Specification within session	NO	

PC - Control of Personal-Computer Access Method

This Natural profile parameter is for:

- mainframes
- UNIX/OpenVMS

It only applies if Natural Connection is installed.

This parameter determines whether support of the personal-computer access method is to be provided using Natural Connection.

Possible settings	ON	Personal-computer support is enabled. The Natural statements READ PC FILE and WRITE PC FILE can be used (for uploading and downloading respectively); see UPLOAD PC FILE and DOWNLOAD PC FILE in the Natural Connection documentation.
	OFF	No personal-computer support is to be provided.
Default setting	OFF	
Dynamic specification	YES	
Specification within session	YES	

Mainframe-specific Information

In addition to enabling personal computer support, you can set the following options:

Option	Explanation
PC=(ON,NAM)	Field names are sent when uploading/downloading data (this is the default).
PC=(ON,NONAM)	No field names are sent when uploading/downloading data.
PC=(ON,NEW)	If Entire Connection Version 4.1.1 or above is installed, • terminal models 3, 4 and 5 • data formats date and time are supported for data transfer (this is the default).
PC=(ON,OLD)	A runtime error will occur if transfer of data formats date or time is attempted. Terminal models 3, 4 and 5 will not use the full size of the terminal buffer; terminal model 2 (screen size 24x80) will be used instead. Use OLD to avoid conflicts when working with older versions of Entire Connection.

The NONAM/NAM and OLD/NEW options can be combined with one another.

Example:

```
PC=(ON,NONAM,OLD)
```

 Copyright © Software AG 2002

See the **Natural Connection documentation** for further information.

If PC is set to ON, the system variable *DEVICE will always contain the value "PC".

The files used for the PC access method have to be defined with the macros NTPRINT and NTWORK or the profile parameters PRINT, WORK and HCAM.

PCNTRL - Print-Control Characters

This Natural profile parameter is for mainframes only.

It determines the line-advance characters for printing which are inserted in column 0 of each print line.

Possible settings	Any character string	This parameter can be specified in character or hexadecimal format (see below).
Default setting	Siemens Environments	`X'40414243444546474849 4A4B4C4D4E4F'`
	IBM Environments	The default setting (according to ASA standard settings) is: `' 0-'` which means that a blank causes a line advance of 1 line, "0" of 2 lines and "-" of 3 lines. In any IBM environment, **do not change the default setting** of this parameter.
Dynamic specification	YES	
Specification within session	NO	

 Copyright © Software AG 2002

PD - Number of Pages Recorded by NATPAGE

This Natural profile parameter is for:

- mainframes
- UNIX/OpenVMS

It specifies the maximum number of pages (screens) which can be stored at the same time in the Natural system file (FUSER) with the NATPAGE screen-paging utility as described in the section Debugging and Monitoring.

Possible settings	0 - 255	Maximum number of pages (screens).
Default setting	50	
Dynamic specification	YES	
Specification within session	YES	Within a Natural session, the profile parameter PD can be overridden by the **session parameter** PD.

If the number of stored screens exceeds the setting of PD, wrap-around technique is used for the system file, which means that the oldest page is overwritten.

PLOG - Logging of Dynamic Parameters

This Natural profile parameter is for mainframes and only applies in batch mode and under TSO.

It enables you to print a list of all profile parameters that were specified dynamically at the start of the session. This may be useful to ascertain which dynamic profile parameters were actually used, particularly if profile parameters like PROFILE or SYS are specified, which in turn "contain" other profile parameters (for a PROFILE or SYS parameter, the entire string of profile parameters activated by it is listed).

Possible settings	ON	At session start, a list of the dynamically specified profile parameters and their settings is written to the output dataset CMPLOG. (If CMPLOG is not available, the list is written to the standard output dataset, CMPRINT.)
	OFF	No list of dynamic profile parameters is written.
Default setting	OFF	
Dynamic specification	YES	See below.
Specification within session	NO	

When specified dynamically, the PLOG parameter applies to all subsequent dynamic profile parameters until the next PLOG specification. This allows you to exclude individual parameters from being printed, for example, if their settings contain passwords or other sensitive information that should not be printed.

All dynamic parameters which are specified between a PLOG=OFF specification and a subsequent PLOG=ON specification are not printed.

 Copyright © Software AG 2002

PLUGIN - Enable the Natural Plug-In Components

This Natural profile parameter is for mainframes only.

It is used to enable the NATTURBO performance components.Currently there are 2 components: BP and PL. The parameter is required when you are using the Natural Turbo Performance Plug-In.

Possible settings	ON	Activates all plug-in components and is equivalent to PLUGIN=(BP,PL).
	OFF	Deactivates all plug-in components and is equivalent to PLUGIN=(NOBP,NOPL).
	BP	Activates the enhanced buffer pool manager, buffer pool cache and the enhanced SYSBPM application.
	PL	Activates the enhanced program loader. The enhanced program loader uses the Adabas multi fetch option to significantly reduce the number of Adabas calls necessary to load a Natural object. The more objects are loaded into the Natural buffer pool and the larger the objects are, the higher the saving of Adabas calls. If parameter PLUGIN=PL is specified, the value specified for the CFWSIZE parameter (default is 32 KB) is automatically increased by 16 KB. Adapt your threadsize accordingly, if need be.
	NOBP	Intended to be used for dynamic specification to reset the PLUGIN setting of the default Natural parameter module NATPARM.
	NOPL	
Default setting	OFF	
Dynamic specification	YES	
Specification within session	NO	

The plug-in components can be activated or deactivated independently.

The PLUGIN parameter can be specified as a single setting or as a setting list enclosed in brackets, e.g. PLUGIN=(BP,NOPL).

Natural sessions using the default buffer pool manager can access only those buffer pools which were initialized with PLUGIN=NOBP.

Natural sessions using the enhanced buffer pool manager can access any buffer pool.

In multi-user environments using a local buffer pool (for example, under CICS), the PLUGIN setting of the first Natural session accessing this buffer pool determines whether the buffer pool is initialized with the default, i.e. the earlier version, (NOBP) or the enhanced buffer pool manager (BP); this means, with PLUGIN=BP, the buffer pool is initialized with hash table support.

PM - Print Mode

This Natural profile parameter is for:

- mainframes
- UNIX/OpenVMS and Windows

It indicates how fields are to be printed or displayed.

PM - Mainframes

Possible settings	C, P, I, R, or combinations CI, CR, PI, PR	**PM=C** An alternative character set is to be used. It can be defined by the profile parameters TAB1 and TAB2. **PM=P** The primary (standard) character set is to be used. **PM=I** This indicates inverse direction (for example, for use in Middle Eastern countries). See the module NATPM in the Natural source library for further information. **PM=R** This resets the PM=I setting to normal (left to right) display direction.
Default setting	PR	
Dynamic specification	YES	
Specification within session	YES	Within a Natural session, the profile parameter PM can be overridden by the **session parameter** PM.

PM - UNIX/OpenVMS and Windows

Possible settings	I	Indicates that field settings are to be displayed in inverse direction; that is, from right to left (for example, for use in Middle Eastern countries).
	R	Resets the "PM=I" setting to normal (left to right) display direction. "PM=R" can only be specified dynamically.
Default setting	R	
Dynamic specification	YES	
Specification within session	YES	

 Copyright © Software AG 2002

POS22 - Version 2.2 Algorithm for POS System Function

This Natural profile parameter is for mainframes only.

It can be used to assure that fields are marked / referenced correctly.

Possible settings	ON	The old Version 2.2 algorithm will be used and the correct field will be marked/referenced **Note**: Set POS22=ON only if you execute objects using the POS function which were compiled with Natural Optimizer Compiler Version 2.2. For any other objects, the change in the internal POS algorithm does *not* lead to different results!
	OFF	The algorithm introduced with Natural Version 2.3 is used.
Default setting	OFF	
Dynamic specification	YES	
Specification within session	NO	

As of Natural Version 2.3, the internal algorithm for the computation of the system function POS (internal field identification) is different. As a result, if you execute with Version 2.3 or higher programming objects which use the POS function in conjunction with the MARK option of an INPUT or REINPUT statement or with the system variable *CURS-FIELD **and** which were compiled with Version 2.2 of the Natural Optimizer Compiler, the wrong field may be marked/referenced.

This parameter will be available only for a limited period of time to allow you a smooth transition from Version 2.2. It will be removed again with a subsequent release of Natural.

In a **Sysplex environment**, or if the Natural thread size makes buffer reallocations necessary, POS22=ON cannot always be guaranteed to yield the desired results. In these cases, the objects concerned should be recataloged in the new version of the Natural Optimizer Compiler to ensure correct POS result.

 PRGPAR - Data to be Passed to Program Receiving Control at Termination

PRGPAR - Data to be Passed to Program Receiving Control at Termination

This Natural profile parameter is for UNIX/OpenVMS and Windows platforms only.

It specifies data to be passed to the program that receives control when Natural terminates (see also the PROGRAM parameter).

Possible settings	any valid character string	Data to be passed.
Default setting	blank	No data is passed.
Dynamic specification	NO	
Specification within session	NO	

 Copyright © Software AG 2002

PRINT - Printer Assignments

This Natural profile parameter is for mainframes only.

It corresponds to the macro NTPRINT in the parameter module. It can be used to specify dynamically the same options which you can specify statically with the macro NTPRINT.

Possible settings	see macro	
Default setting	none	
Dynamic specification	YES	This parameter can only be specified dynamically.
Specification within session	NO	

See also FAMSTD - Overwriting of Print and Work File Access Method Assignments.

PROFILE - Activate Dynamic Parameter Profile

This Natural profile parameter is for mainframes only.

When you invoke Natural with dynamic profile parameters, instead of having to specify a whole string of individual parameters each time you invoke Natural, you can specify the string of parameters once, store this string under a profile name and then invoke Natural with only one dynamic parameter as follows:

`PROFILE=`*`profile-name`*

The parameters defined under this profile are passed to Natural as dynamic profile parameters.You create and maintain these profiles with the utility SYSPARM (described in the Natural Utilities for Mainframes documentation).

Possible settings	1-8 characters	Profile-name. Or special options, see below.
Default setting		
Dynamic specification	YES	
Specification within session	NO	

Special Options

In addition, the PROFILE parameter provides the following special options:

PROFILE=AUTO	Natural takes the current TP user ID (as contained in the system variable *INIT-USER) as profile name, which means that the profile defined under the name corresponding to that ID is used. If no such profile is found, a profile named "AUTO" is used instead (if available). You can define such an "AUTO" profile as default profile for users without individual profiles.
PROFILE=TERMINAL	Natural takes the current terminal ID (as contained in the system variable *INIT-ID) as profile name, which means that the profile defined under the name corresponding to that ID is used.
PROFILE=PROGRAM	Natural takes the name of the program currently executing as Natural (as contained in the system variable *INIT-PROGRAM) as profile name, which means that the profile defined under this name is used.

By default, the profile is read from the current FNAT system file.

To read it from a different system file, you can specify the desired database ID, file number, password and cipher code after the *profile-name* (or after one of the above special options) as follows:
PROFILE=(*profile-name*,*dbid*,*fnr*,*password*,*cipher-code*)

To ensure that all profile parameters are read from the same system file (other than FNAT), specify **PROFILE=(,***dbid***,***fnr***)** in the parameter module.

If the PROFILE parameter is specified within a parameter module, it is evaluated **after** the other parameters in the parameter module, but **before** any dynamically specified profile parameters are evaluated; this means that parameters specified within the profile can be overridden by individually specified dynamic parameters.

 Copyright © Software AG 2002

To restrict the use of a profile, you can use the USER profile parameter.

Unlike other parameters, a PROFILE parameter specification cannot be overwritten by another PROFILE. So you can have multiple parameter profiles which are evaluated all in a sequence.

The PROFILE parameter cannot be used with ADARUN MODE=SINGLE.

PROGRAM - Non-Natural Program Receiving Control after Termination

This Natural profile parameter is for:

- mainframes (except VM/CMS)
- UNIX/OpenVMS and Windows

Mainframes

This parameter specifies a non-Natural back-end program which is to receive control after the termination of the Natural session.

Possible settings	1-8 characters	Non-Natural back-end program.
Default setting	none	
Dynamic specification	YES	
Specification within session	YES	

It can also be set dynamically from within a Natural program by calling the subroutine CMPGMSET which is provided in the library SYSEXTP.

Data for the program specified with the PROGRAM parameter can be supplied with the TERMINATE statement (described in the Natural Statements documentation).

For the conventions of calling non-Natural back-end programs, see Back-End Program Calling Conventions.

CICS-specific information:

In addition to back-end programs, the Natural CICS interface also supports back-end transactions which may be specified via ’RET=XXXX’ or ’STR=XXXX’ instead of a program name, with XXXX being a valid CICS transaction ID.

- RET=XXXX indicates that control has to be passed to CICS together with a return transaction ID by a CICS RETURN TRANSID (’XXXX’) command.
- STR=XXXX indicates that a new transaction has to be started by a CICS START TRANSID (’XXXX’) TERMID (*INIT-ID), before relinquishing control via a CICS RETURN command.

UNIX/OpenVMS and Windows

This parameter specifies the non-Natural program which is to receive control after the termination of the Natural session.

Possible settings	1-12 characters	Non-Natural program
Default setting	none	
Dynamic specification	YES	
Specification within session	NO	

 Copyright © Software AG 2002

Data for the program specified with the PROGRAM parameter can be supplied with the TERMINATE statement.

PROX - Specify URL of Proxy Server

This parameter is for UNIX/OpenVMS only.

This Natural profile parameter specifies the URL of the (Intranet) proxy server through which all requests have to be routed (optional).

Possible settings	URL of proxy server
Default setting	None
Dynamic specification	NO
Specification within session	NO

 Copyright © Software AG 2002

PROXPORT - Specify Port Number of Proxy

This parameter is for UNIX/OpenVMS only.

This Natural profile parameter specifies the port number of the proxy, if any is set.

Possible settings	Port number of proxy, 4 characters at maximum.
Default setting	80
Dynamic specification	NO
Specification within session	NO

PS - Page Size for Natural Reports

This Natural profile parameter is for:

- mainframes
- UNIX/OpenVMS and Windows

It indicates the maximum number of lines per page to be used for Natural reports created with the DISPLAY or WRITE statement.

Possible settings	0, or 5 - 250	Maximum number of lines per page.
Default setting	0	The output device page size is to be used.
Dynamic specification	YES	
Specification within session	YES	Within a Natural session, the profile parameter PS can be overridden by the **session parameter** PS.

This parameter, together with the profile parameter LS, defines the amount of page buffer space acquired during Natural initialization.

Note for mainframe users:
The product PS * LS must not exceed 32 KB.

 Copyright © Software AG 2002

PSEUDO - CICS Pseudo-Conversational Mode

This Natural profile parameter is for mainframes only.

It controls the mode of operation under CICS. When Natural is executing under control of the TP monitor CICS, two modes are possible: conversational and pseudo-conversational.

Possible settings	ON	In pseudo-conversational mode, a Natural session is a sequence of different transactions. After each output to the terminal, all Natural work areas are saved and the transaction is terminated. When the user responds to a message by pressing ENTER (or any other input key), a new transaction is initiated. The Natural work areas are restored, the terminal input is read and the Natural session is continued. The transaction identification of each following transaction can be set dynamically by calling the subroutine CMTRNSET, which is provided in the library SYSEXTP.
	OFF	In conversational mode, a Natural session is one transaction, which is active for as long as the Natural session is active.
Default setting	ON	
Dynamic specification	YES	
Specification within session	NO	

For more information, refer to Natural under CICS, section TYPE - Thread Type for Group in the Natural TP Monitor Interfaces documentation.

RCA - Resolve Addresses of Static Non-Natural Programs

This Natural profile parameter is for mainframes only.

It controls the "dynamic" linking of static non-Natural programs to the Natural nucleus during initialization of the Natural session.

Possible settings	ON	At Natural startup, the list of all static non-Natural programs to be linked to Natural is scanned and a load request is issued for all modules whose addresses are unresolved. If a load request fails, no error message is issued. The use of RCA=ON is *not* recommended, as it causes a lot of processing overhead at Natural startup.
	OFF	No dynamic linking of static non-Natural programs is performed.
	list of names	If "RCA=*name-list*" is specified the list of static non-Natural programs to be linked to Natural is extended by the specified name list. A load request is issued for these modules even if they are already linked. In this way, it is possible to replace linked non-Natural programs. If a load request fails, an error message is issued. If more than one name is specified, each must be separated from the next by a comma and the list must be enclosed within parentheses as shown below: RCA=(PROGRAM1,PROGRAM2,PROGRAM3)
Default setting	OFF	
Dynamic specification	YES	
Specification within session	NO	

Static non-Natural programs have to be defined for being linked to Natural either internally (by using the macro NTINV within the modules NATPARM and NATCONFG) or externally (by using the profile parameter CSTATIC).

If the external name of the non-Natural program is different from the internal one (as used by the CALL statement), you can use either the dynamic parameter RCALIAS or the macro NTALIAS to define which external name is to be used for the load request.

The following platform-specific requirements apply:

Platform:	**Comment:**
CICS	A PPT entry has to be defined to allow the load request for a non-Natural program. Static non-Natural programs are called via standard linkage conventions rather than EXEC CICS LINK requests.

 Copyright © Software AG 2002

 RCALIAS - External Name Definition for Non-Natural Programs

RCALIAS - External Name Definition for Non-Natural Programs

This Natural profile parameter is for mainframes only.

It corresponds to the macro NTALIAS, which can be specified in the Natural parameter module NATPARM. It only applies if the profile parameter RCA is set to "ON" or to "*name-list*".

RCALIAS is used to define the external names of static non-Natural programs which are loaded for dynamic linking during the initialization of a Natural session.

Possible settings	list of name pairs	The first name of a name pair defines the internal name of a non-Natural program, which must also be defined by the parameter CSTATIC or RCA. The second name defines the corresponding external alias name for the load request. Example: `RCA=(PROGRAM1,PROGRAM2),RCALIAS=(PROGRAM1,ALIAS1,PROGRAM2,ALIAS2)`
	OFF	The parameter is inactive.
Default setting	OFF	
Dynamic specification	YES	This parameter can only be specified dynamically.
Specification within session	NO	

 RCFIND - Handling of Response Code 113 for FIND Statement

RCFIND - Handling of Response Code 113 for FIND Statement

This Natural profile parameter is for mainframes only.

It determines the action to be taken if Adabas Response Code 113 (requested ISN not found) is returned during the execution of a FIND statement processing loop.

Possible settings	ON	Response Code 113 causes the program to be terminated.
	OFF	Response Code 113 will be ignored, and processing of the FIND loop will continue by reading the next record.
Default setting	ON	
Dynamic specification	YES	
Specification within session	NO	

 Copyright © Software AG 2002

RCGET - Handling of Response Code 113 for GET Statement

This Natural profile parameter is for mainframes only.

It determines the action to be taken if Adabas Response Code 113 (requested ISN not found) is returned during the execution of a GET statement.

Possible settings	ON	Response Code 113 causes the program to be terminated.
	OFF	Response Code 113 will be ignored, the system variable *ISN will be set to "0", and processing will continue.
Default setting	ON	
Dynamic specification	YES	
Specification within session	NO	

RDACTIVE - Activate Remote Debugger

This Natural profile parameter is for UNIX/OpenVMS and Windows platforms only.

It specifies whether a remote debugger on a Windows NT computer is to be used or not. RDACTIVE is only relevant if the **system command** DEBUG has been used. It is used in combination with the profile parameters RDNODE (where the relevant node name is specified) and PDPORT (where the port number is specified).

Possible settings	ON	Remote debugging is enabled. If you specify RDACTIVE=ON on a DCOM or RPC server, the remote debugging session is opened automatically.
	OFF	UNIX/OpenVMS: No debugging is possible.
		Windows: Remote debugging is not possible. Local debugging can be performed.
Default setting	OFF	
Dynamic specification	YES	
Specification within session	NO	

 Copyright © Software AG 2002

RDCEXIT - Define Natural Data Collector User Exits

This Natural profile parameter is for mainframes only.

It is used to define Natural Data Collector user exits and, optionally, a work area size for each exit. If linked, the exit gets control from the Natural Data Collector at certain points within Natural. Specific session information is passed to the exits.

Possible settings	A list of user exit names with a work area size for each user exit.	With RDCEXIT, any exit names can be specified. In the Natural parameter module NATPARM, the exit names are automatically added to the CSTATIC profile parameter's setting list.
Default setting	none	
Dynamic specification	YES	If RDCEXIT is specified dynamically, the exits must be defined in the CSTATIC or RCA profile parameters (RCA can also be specified dynamically). Optionally, the size of the exit work area can be specified after the exit name. Possible settings: 400 to 32760. The default is 400. Example: `RDCEXIT=(MYEXIT,2000,RDCEX1)`
Specification within session	NO	

For details, refer to SYSRDC Utility in the section Debugging and Monitoring.

RDCSIZE - Size of Buffer for the Natural Data Collector

This Natural profile parameter is for mainframes only.

It determines the for the Natural Data Collector of the SYSRDC utility. In addition, it controls the trace recording function of the data collector.

Possible settings	0, or 2 - 128	Buffer size in KB. To activate the data collector (without trace recording), you specify RDCSIZE=2. To also activate the trace recording, you have to set RDCSIZE to a setting greater than "2". If the requested space is not available, the Natural Data Collector of the SYSRDC utility cannot be used.
Default setting	0	Deactivates the data collector.
Dynamic specification	YES	
Specification within session	NO	

For details, refer to SYSRDC Utility in the section Debugging and Monitoring.

 Copyright © Software AG 2002

RDNODE - Remote Debugger Node Name

This Natural profile parameter is for UNIX/OpenVMS and Windows platforms only.

It specifies the node name of the Windows NT computer on which a remote debugger is to be called.

Possible settings	character string	Node name.
Default setting	none	
Dynamic specification	YES	
Specification within session	NO	

RDPORT - Remote Debugger Port

This Natural profile parameter is for UNIX/OpenVMS and Windows platforms only.

It specifies the port number on which a remote debugger on a Windows NT computer has been installed.

Possible settings	0 - 9999	Port number. By default, the debugger service is installed on the Windows NT computer's Port 2500. You can therefore leave the OpenVMS or UNIX computer's RDPORT setting at the default. If, however, Port 2500 on Windows NT has been reserved for another service, and a different port number has been specified, you must change RDPORT accordingly.
Default setting	2500	
Dynamic specification	YES	
Specification within session	NO	

 Copyright © Software AG 2002

RDS - Define Remote Directory Server

This Natural profile parameter is for UNIX/OpenVMS and Windows platforms only.

Note for mainframes:
Use the keyword parameter RDS of the NTRPC macro for static specification, or the subparameter with the same name of the profile parameter RPC.

RDS is specified on the client side only.

It allows you to define up to 10 remote directory servers. For each remote directory server, you specify up to 5 subparameters.

Possible settings	5 subparameters	The *server name* (1 - 8 characters). There is no default, the value must be specified.
		The *server node* (1 - 8 characters). There is no default, the value must be specified.
		The name of the *subprogram*, titled CALLNAT, (1 - 8 characters) to be used as interface (default is RDSSCDIR).
		A *logon indicator*. If nothing is specified, blank is the default. L The client initiates a Natural logon to the server with the library name of the current library on the client. blank No server logon will be executed. Note for Windows platforms: instead of "L", check the selection box.
		The *transport protocol* to be used (ACI or CSCI). ACI is the default. CSCI is supported only for OpenVMS.
Default setting	none	Subparameter defaults, see above.
Dynamic specification	YES	See below.
Specification within session	NO	

For dynamic specification the syntax is as follows.

Using 1 server:
RDS=(*server-name*,*server-node name*,*subprogram*,*logon-indicator*,*transport-protocol-name*)

Using 2-10 servers:
RDS=((*server-name*,*server-node name*,subprogram,*logon-indicator*,*transport-protocol-name*)(*server-name*,*server-node name*,subprogram,*logon-indicator*,*transport-protocol-name*)...(*server-name*,*server-node name*,subprogram,*logon-indicator*,*transport-protocol-name*))

For further information see the Natural Remote Procedure Call documentation.

READER - VSE/ESA System Logical Units for Input

This Natural profile parameter is for mainframes only.

It indicates the VSE/ESA system logical units which are to be used by Natural for input.

Possible settings	READER=(*n*,*device*,...)	*n* is "0" for CMSYNIN and "1" for CMOBJIN *device* is either "SYSRDR" or "SYSIPT"
Default setting	READER=(0,SYSRDR,1,SYSIPT)	By default, the primary input stream (CMSYNIN) is read from SYSRDR and the input stream (CMOBJIN) is read from SYSIPT (if required). If CMSYNIN or CMOBJIN are disk or tape files, the associated READER subparameter is ignored.
Dynamic specification	YES	
Specification within session	NO	

This overwriting of a system logical unit number only applies if the relevant file is a card file. If CMSYNIN or CMOBJIN are disk or tape files, the associated READER subparameter is ignored.

 Copyright © Software AG 2002

RECAT - Dynamic Recataloging

This Natural profile parameter is for:

- mainframes
- UNIX/OpenVMS and Windows

Mainframes

This parameter indicates the action to be taken if Natural detects an inconsistency in the global data area definition as defined in the program currently being executed; that is, the global data area in the program does not correspond to the definition of the global data area currently in use.

Possible settings	ON	An error message is issued if an inconsistency concerning a Version 2.2 program and/or global data area is detected. Natural automatically adjusts the object and disables the **system commands** CATALOG, PURGE and SAVE. **Note:** If the adjusted object invokes an object from a steplib library that also has to be adjusted, the object from the steplib library will be copied to the library of the invoking object.
	OFF	Natural issues an error message.
Default setting	OFF	
Dynamic specification	YES	
Specification within session	NO	

This profile parameter only applies to Natural objects of Version 2.3 and above.

UNIX/OpenVMS and Windows

This parameter indicates the action to be taken if Natural detects an inconsistency in the global data area definition as defined in the program currently being executed.

Possible settings	ON	Natural issues an error message and disables the **system commands** CATALOG, PURGE and SAVE.
	OFF	Natural issues an error message only.
Default setting	OFF	
Dynamic specification	YES	
Specification within session	NO	

If the RECAT parameter has been set to ON, and an object exists in both source and cataloged form, then the source and the cataloged object cannot be processed independently. In order to ensure consistency between the source and the cataloged object, Natural disables the system command CATALOG (also when invoked via CATALL). In addition, the system commands PURGE and SAVE are disabled for a source for which a corresponding cataloged object exists.

Note:
Only objects satisfying the criteria for a particular command (such as PURGE) will be displayed in the corresponding selection box.

 Copyright © Software AG 2002

REINP - Issue Internal REINPUT Statement for Invalid Data

This Natural profile parameter is for:

- mainframes
- UNIX/OpenVMS and Windows

By default, Natural automatically issues an internal REINPUT statement if invalid data have been entered in response to an INPUT statement. With this parameter, you can switch this mechanism off. This will allow you to handle such input errors yourself in your application.

Possible settings	ON	An internal REINPUT statement is issued when invalid data have been entered.
	OFF	An internal REINPUT statement is not issued when invalid data have been entered.
Default setting	ON	
Dynamic specification	YES	
Specification within session	YES	Within a Natural session, the profile parameter REINP can be overridden by the **session parameter** REINP.

Copyright © Software AG 2002

RELO - Storage Thread Relocation

This Natural profile parameter is for mainframes only.

It controls the relocation of the Natural thread after a terminal I/O in a thread environment (CICS, Com-plete, IMS/TM, UTM and Natural Server under OS/390).

Possible settings	ON	The Natural thread and all the buffers contained therein can be relocated to another storage area if the original storage area has been occupied by another user after a terminal I/O.
	OFF	No relocation is performed. The Natural thread and all the buffers therein remain located at the same virtual address after the terminal I/O. This setting applies to CICS and Com-plete only. In all other thread environments, Natural cannot guarantee that the thread remains located at the same address. **Note for CICS:** When using TYPE=GETM threads under CICS, RELO=OFF has the same effect as the PSEUDO=OFF setting of the PSEUDO Parameter. See also TYPE - Thread Type for Group in the section Natural under CICS in the TP Monitor Interfaces documentation.
	FORCE	This will force a relocation of the Natural thread and all the buffers contained therein to another storage area. This can be useful for testing purposes in some environments. This setting applies to CICS, Com-plete and IMS/TM only
Default setting	ON	
Dynamic specification	YES	
Specification within session	NO	

 Copyright © Software AG 2002

RFILE - File for Recordings

This Natural profile parameter is for mainframes only.

It determines where recordings (that is, the data recorded by the Recording function) are stored.

Possible settings	SPAD	Recordings will be stored in the scratch-pad file. (If no scratch-pad file is defined, the recordings will be stored in the system file FUSER.)
	FUSER	Recordings will be stored in the system file FUSER.
	FNAT	Recordings will be stored in the system file FNAT.
Default setting	SPAD	
Dynamic specification	YES	
Specification within session	NO	

For details on the Recording function, see the NATPAGE utility in the section Debugging and Monitoring.

RI - Release ISNs

This Natural profile parameter is for mainframes only.

It indicates whether ISNs for records which were read and placed in hold status but were not updated are to be retained in hold status.

Possible settings	ON	Natural releases the ISN of each record which has been placed in hold status but was not updated (for example because the record was rejected as a result of a WHERE clause or an ACCEPT/REJECT statement). This reduces the number of ISNs which are contained in the hold queue. This may, however, cause additional performance overhead as an Adabas call is required for each ISN released.
	OFF	The ISN of each record which has been placed in hold status is *not* released until the end of the transaction.
Default setting	OFF	
Dynamic specification	YES	
Specification within session	NO	

In nested processing loops, a record which due to RI=ON is released in an inner processing loop is no longer kept in hold status for any outer loop.

 Copyright © Software AG 2002

RJESIZE - Initial Size of NATRJE Buffer

This Natural profile parameter is for mainframes only.

With the Natural utility NATRJE (described in the Natural Utilities for Mainframes documentation), JCL jobs can be collected and then submitted all at once. RJESIZE is used to set the initial size of the buffer which holds the JCL jobs before they are submitted.

Possible settings	0 - 256	Buffer size in KB. If the initial size is not sufficient, Natural automatically increases (repeatedly if necessary) the buffer size by 8 KB. To disable the NATRJE utility, set RJESIZE=0.
Default setting	8	
Dynamic specification	YES	
Specification within session	NO	

RM - Retransmit Modified Fields

This Natural profile parameter is for mainframes only.

Some TP monitors translate input data automatically to upper-case characters. As Natural's screen optimization only retransmits modified data back to the screen, the TP-monitor translation may cause input for a field which has been modified not to be retransmitted.

Possible settings	ON	Natural always sends back all modified fields.
	OFF	Natural sends back modified fields only if they have been changed.
Default setting	OFF	
Dynamic specification	YES	
Specification within session	NO	

 Copyright © Software AG 2002

ROSY - Read-Only Access to System Files

This Natural profile parameter is for:

- mainframes
- UNIX/OpenVMS and Windows

It disables modifications on the Natural system files FNAT, FUSER and FSEC.

Possible settings	ON	No data can be written to, modified on or deleted from the system files. Natural issues an error message instead of performing any action that would modify any of these system files.
	OFF	Data can be written to, modified on and deleted from the system files.
Default setting	OFF	
Dynamic specification	YES	
Specification within session	NO	

For Mainframes the following applies:

If your system files are specified as read-only (ROSY=ON), the Natural functions SETUP/RETURN, Recording, NATPAGE and invoking of helproutines cannot be used, because they write data to the Natural system files FNAT and/or FUSER.

Therefore, it is recommended that you allocate and use a so-called scratch-pad file to hold these temporary data. The scratch-pad file is optional and must be defined as recoverable by using the macro NTFILE or the parameter LFILE. The above functions then write their data to this file instead of FNAT/FUSER.

With ROSY=OFF, a scratch-pad file should also be defined if you use the Recording and NATPAGE functions with database transaction logic, as that might lead to unpredictable results with FNAT/FUSER.

Copyright © Software AG 2002

RPC - Remote-Procedure-Call Settings

This Natural profile parameter is for mainframes only.

With this parameter you can specify subparameters which affect the handling of Natural RPC. These subparameters correspond to the keyword parameters of the macro NTRPC in the parameter module. They can be used to specify dynamically the same options which can be specified statically with the macro NTRPC.

Possible settings	subparameters	see macro NTRPC.
Default setting	none	
Dynamic specification	YES	This parameter can only be specified dynamically.
Specification within session	NO	

The entire string of the RPC parameter must be enclosed in parentheses, for example:
`RPC=(RPCSIZE=80,SRVNAME=MYSERV,SERVER=ON,DFS=(SRV2,NODE1,,ACI))`

 Copyright © Software AG 2002

RPCSIZE - Size of Buffer Used by Natural RPC

This Natural profile parameter is for UNIX/OpenVMS and Windows platforms only. (It was formerly called SIZE.)

Note for mainframes:
Use the keyword parameter RPCSIZE of the NTRPC macro for static specification, or the subparameter with the same name of the profile parameter RPC.

It determines the size of the buffer used by Natural RPC.

RPCSIZE can be specified on both the client and the server side.

Possible settings	0 - 16384	Buffer size *nnn* in KB.
Default setting	0	Natural RPC cannot be used.
Dynamic specification	YES	
Specification within session	NO	

For further information see the Natural Remote Procedure Call documentation.

RUNSIZE - Size of Runtime Buffer

This Natural profile parameter is for mainframes only.

It sets the size of the Natural runtime buffer.

Possible settings	10 - 64	Buffer size in KB.
Default setting	16	
Dynamic specification	YES	
Specification within session	NO	

The Natural runtime buffer contains information on the following items:

- defined STEPLIBS,
- the file translation table (profile parameter TF),
- log information of the most recent command,
- the environment stack (for user settings),
- active global data areas,
- invoked subroutines (subroutine name and object name),
- invoked objects (address in the buffer pool for a fast location).

If the specified size of the runtime buffer is exceeded by a Natural user, the size for the invoked objects information is decreased accordingly. However, when this size decreases, the number of possible buffer-pool fast locations decreases, too; if it is about to become 0, an error message is issued.

 Copyright © Software AG 2002

SA - Sound Terminal Alarm

This Natural profile parameter is for:

- mainframes
- UNIX/OpenVMS and Windows

It indicates whether the terminal alarm feature is to be used.

Possible settings	ON	The terminal alarm is sounded each time the user is prompted for input by Natural. Mainframes: The use of this feature requires that the terminal alarm hardware feature has been installed for the terminal.
	OFF	No terminal alarm is used for input prompting.
Default setting	OFF	
Dynamic specification	YES	
Specification within session	YES	Within a Natural session, the profile parameter SA can be overridden by the **session parameter** SA.

SCTAB - Scanner Character-Type Table

This Natural profile parameter is for mainframes only.

It corresponds to the NTSCTAB macro in the parameter module. It can be used to specify dynamically the same options which you can specify statically with the macro NTSCTAB.

Possible settings	See NTSCTAB macro.	
Default setting	none	
Dynamic specification	YES	This parameter can only be specified dynamically.
Specification within session	NO	

 Copyright © Software AG 2002

SD - Time Delay between Two Screens

This Natural profile parameter is for UNIX/OpenVMS and Windows platforms only.

It delays the time related to screen output display. This is the time delay between two screens during a non-conversational write operation (see the terminal command %N in the Natural Reference documentation).

Possible settings	0 - 100	The unit for the specified setting is a tenth of a second, which means that "SD=10" leads to a delay of one second.
Default setting	0	
Dynamic specification	NO	
Specification within session	NO	

SENDER - Screen Output Destination for Asynchronous Processing

This Natural profile parameter is for mainframes and only applies under CICS, Com-plete, IMS/TM and UTM.

It specifies the destination where output from an asynchronous application is to be displayed. The destination specified applies to hardcopy output and primary reports.

Possible settings	1 to 8 characters	Output destination, e.g. printer.
Default setting	none	
Dynamic specification	YES	
Specification within session	NO	

Any additional reports are sent to the destinations specified with the DEFINE PRINTER statement (just as in a synchronous online session).

The following platform-specific characteristics apply:

Platform:	**Comment:**
CICS	The SENDER parameter specifies the CICS transient data (TD) destination and the terminal or printer for terminal output from asynchronous sessions. If the specified destination does not exist, the session output is sent to the specified terminal or printer. If the specified terminal or printer does not exist either, the session terminates abnormally. The default terminal output format for asynchronous sessions is a 3270 data stream. If the SENDER terminal specification is not a 3270 device, the Natural application must switch to the correct terminal type before the first output statement (for example, by specifying SET CONTROL ’T=PRNT’ for a printer or by starting with profile parameter TTYPE=PRNT). If you are routing all output to a (spool) destination, such as CSSL, the Natural application must be switched to line mode, for example by specifying SET CONTROL ’T=*xxxx*’ or by starting with profile parameter TTYPE=*xxxx*, where *xxxx* is BTCH or ASYL. In this case, two other profile parameters are relevant: EJ and INTENS. If you set EJ=ON, all lines are routed with a leading ASA control character. With EJ=OFF, there is no leading ASA control character. INTENS should be set to "1", particularly if you have set EJ=OFF. For other CICS-specific functionality, see Asynchronous Natural Sessions under CICS (in the Natural TP Monitor Interfaces documentation).
IMS/TM	The SENDER specifies the default LTERM. This LTERM is always used when no other printer has been specified. You should always dynamically define a SENDER parameter in the NIIBOOT module. This is important when Natural tries to output error messages when starting a session: if no SENDER parameter is specified, there is no valid LTERM and NATIMS terminates the session.
UTM	The SENDER specifies the ID for the initialization of an asynchronous transaction; that is, the ID which identifies the transaction as asynchronous. If output from an asynchronous transaction is to be printed, the setting specified with the SENDER parameter also identifies the printer on which the output is to be printed.

 Copyright © Software AG 2002

 SENDER - Screen Output Destination for Asynchronous Processing

For further information, see also the profile parameter OUTDEST and synchronous Processing.

SERVER - Start Natural Session as an RPC Server Session

This Natural profile parameter is for UNIX/OpenVMS and Windows platforms only.

Note for mainframes:
Use the keyword parameter SERVER of the NTRPC macro for static specification, or the subparameter with the same name of the profile parameter RPC.

It specifies whether or not the Natural session will be started as an RPC server session.

SERVER can be specified on both the client and the server side.

Possible settings	ON	The Natural session will be started as an RPC server session.
	OFF	The Natural session will not be started as an RPC server session.
Default setting	OFF	
Dynamic specification	YES	
Specification within session	NO	

For further information see the Natural Remote Procedure Call documentation.

 Copyright © Software AG 2002

SF - Spacing Factor

This Natural profile parameter is for:

- mainframes
- UNIX/OpenVMS and Windows

It indicates the default number of spaces to be inserted between field settings of columns on Natural reports created using a DISPLAY statement.

Possible settings	1 - 30	Number of spaces.
Default setting	1	
Dynamic specification	YES	
Specification within session	YES	Within a Natural session, the profile parameter SF can be overridden by the **session parameter** SF.

SHELL - Grant Shell Access to Natural User

This Natural profile parameter is for UNIX/OpenVMS only.

It can only be set by Natural administrators. This parameter allows or disallows the shell exit from the Natural Main Menu for specific users.

Be careful when granting shell access to a Natural user, as this might introduce security problems, because the user would then be able to use the underlying features of the operating system.

Possible settings	YES	Shell exit is allowed. If the shell exit is allowed for a user, the name of the shell that will be started for this user can be specified by the administrator. • For OpenVMS users, this shell name must be the name of an executable DCL program, for example, "SPAWN". • For UNIX users, this shell name must be the full path name of an executable UNIX program, for example, "/bin/csh".
	NO	Shell exit is not allowed.
Default setting	NO	
Dynamic specification	NO	
Specification within session	NO	

 Copyright © Software AG 2002

SI - Shift-In Code for Double-Byte Character Set

This Natural profile parameter is for mainframes only.

It is only relevant for Asian countries which use double-byte character sets (DBCS). The parameter is used to specify a shift-in code.

Possible settings	0F	Shift-in code for IBM hardware.
	29	Shift-in code for Fujitsu hardware.
Default setting	none	
Dynamic specification	YES	
Specification within session	NO	

The shift-in code is used to indicate the point at which the code of character representation is shifted from double-byte mode back into normal (single-byte) mode.The beginning of the double-byte character representation (shift-out code) is indicated by the setting defined with the SO parameter.

SKEY - Storage Protection Key

This Natural profile parameter is for mainframes and only applies under Com-plete.

It determines whether Natural runs under the same storage protection key as Com-plete.

Possible settings	ON	Natural runs under the same storage protection key as Com-plete.
	OFF	Natural runs under a different storage protection key than Com-plete.
Default setting	ON	
Dynamic specification	YES	
Specification within session	NO	

See the Com-plete documentation for details on storage protection keys.

 Copyright © Software AG 2002

SL - Source-Line Length

This Natural profile parameter is for mainframes only.

It indicates the number of characters to be interpreted on each Natural source-code line.

Possible settings	20 - 80	In batch mode: the number of characters to be processed on each line in CMSYNIN. Online: the number of characters to be interpreted when using the Natural program editor in EDT mode (as activated with the system command EDT).
Default setting	72	
Dynamic specification	YES	
Specification within session	YES	Within a Natural session, the profile parameter SL can be overridden by the **session parameter** SL.

SM - Programming in Structured Mode

This Natural profile parameter is for:

- mainframes
- UNIX/OpenVMS and Windows

It determines whether or not structured mode must be used.

Possible settings	ON	Structured mode syntax must be used. If specified during the session, GLOBALS SM=OFF is disabled. (The message "Reporting mode not permitted" will be displayed).
	OFF	Programming can be done in either structured mode or reporting mode.
Default setting	OFF	
Dynamic specification	YES	
Specification within session	YES	Within a Natural session, the profile parameter SM=OFF can be overridden by the **session parameter** SM=ON.

With Natural Security, the setting of the Mode option in the library's security profile determines whether the SM profile parameter can be used; see also Programming mode (in the Natural Security documentation).

 Copyright © Software AG 2002

SNAT - Sound a Bell at Syntax Error

This Natural profile parameter is for UNIX/OpenVMS and Windows platforms only.

It is used to sound a bell when the compiler detects a syntax error in a Natural program.

Possible settings	ON	A bell will sound when a syntax error is encountered.
	OFF	No bell will sound in the case of syntax errors.
Default setting	OFF	
Dynamic specification	NO	
Specification within session	NO	

SO - Shift-Out Code for Double-Byte Character Set

This Natural profile parameter is for mainframes only.

It is only relevant for Asian countries which use double-byte character sets (DBCS). The parameter is used to specify a shift-out code.

Possible settings	0E	Shift-out code forIBM hardware.
	28	Shift-out code for Fujitsu hardware.
Default setting	none	
Dynamic specification	YES	
Specification within session	NO	

The shift-out code is used to indicate the point at which the code of character representation is shifted out of normal (single-byte) mode into double-byte mode. The end of the double-byte character representation (shift-in code) is indicated by the setting defined with the SI profile parameter.

 Copyright © Software AG 2002

SORT - Control of Sort Program

This Natural profile parameter is for mainframes only.

It is used to control the sort program used for the processing of SORT statements. With the SORT parameter, you can specify dynamically the same options that you can specify with the NTSORT macro in the parameter module.

Possible settings	see NTSORT macro	For an explanation of the individual options and their possible settings, see the NTSORT macro.
Default setting	see NTSORT macro	
Dynamic specification	YES	This parameter can only be specified dynamically.
Specification within session	NO	

Specify the parameter as follows:

```
SORT=(WRKSIZE=nnn,STORAGE=medium,
       EXT=ON/OFF,EXTNAME=name,EXTOPT=(options,...),EXTEOJ=ON/OFF)
```

SORTSZE - Size of Sort Buffer

This Natural profile parameter is for UNIX/OpenVMS and Windows platforms only.

It indicates the amount of storage to be reserved for use by the sort program. This sort buffer is only allocated when executing a Natural program which contains a SORT statement.

Possible settings	500 - 2048	Buffer size in KB. Increasing the setting leads to faster sort processing, in particular when all data to be sorted fit into the sort buffer.
Default setting	500	
Dynamic specification	YES	
Specification within session	NO	

 Copyright © Software AG 2002

SRVNAME - Name of RPC Server

This Natural profile parameter is for UNIX/OpenVMS and Windows platforms only.

Note for mainframes:
Use the keyword parameter SRVNAME of the NTRPC macro for static specification, or the subparameter with the same name of the profile parameter RPC.

It specifies the name of the RPC server, with which it registers on the node specified with the profile parameter SRVNODE.

SRVNAME is specified on the server side only.

Possible settings	1 - 8 characters	Valid server name.
Default setting	none	
Dynamic specification	YES	
Specification within session	NO	

For further information see the Natural Remote Procedure Call documentation.

SRVNODE - Name of Node

This Natural profile parameter is for UNIX/OpenVMS and Windows platforms only.

Note for mainframes:
Use the keyword parameter SRVNODE of the NTRPC macro for static specification, or the subparameter with the same name of the profile parameter RPC.

It specifies the name of the node upon which an RPC server registers.

SRVNODE is specified on the server side only.

Possible settings	1 - 8 characters	Node name.
Default setting	none	
Dynamic specification	YES	
Specification within session	NO	

For further information see the Natural Remote Procedure Call documentation.

 Copyright © Software AG 2002

SRVUSER - User ID for RPC Server Registry

This Natural profile parameter is for UNIX/OpenVMS and Windows platforms only.

Note for mainframes:
Use the keyword parameter SRVUSER of the NTRPC macro for static specification, or the subparameter with the same name of the profile parameter RPC.

It specifies the user ID needed to register an RPC server on the node specified with the profile parameter SVRNODE. If this specification is omitted, the timestamp will be used.

SRVUSER is specified on the server side only.

Possible settings	1 - 16 characters	User ID. If SRVUSER is set to *NSC and Natural Security is installed, the Natural server uses the current Natural user ID (*USER) and the password defined for this user ID in Natural Security to logon to the node.
Default setting	timestamp	
Dynamic specification	YES	
Specification within session	NO	

For further information see the Natural Remote Procedure Call documentation.

SSIZE - Size of Source Area Allocated by the Editors

This Natural profile parameter is for:

- mainframes
- UNIX/OpenVMS and Windows

Mainframes

It determines the size of the buffer used by the Software AG Editor.

Possible settings	0, or 40 - 128	Buffer size in KB. If SSIZE=0 or if the required space is not available, the Software AG Editor cannot be used.
Default setting	64	
Dynamic specification	YES	
Specification within session	NO	

UNIX/OpenVMS and Windows

This parameter determines the maximum size of the Natural source area, which will be dynamically allocated by the Natural editors.

Possible settings	1 - 100	Maximum size of the Natural source area in MB.
Default setting	1	
Dynamic specification	NO	
Specification within session	NO	

Note:
The maximum size for one Natural source member is 1 MB (independent of SSIZE).

 Copyright © Software AG 2002

STACK - Place Data/Commands on the Stack

This Natural profile parameter is for:

- mainframes
- UNIX/OpenVMS and Windows

It is used to place data/commands on the Natural stack.

Note:
If STACK is used, the parameters DC, HI, IA and ID must not be set to ":" or ";".

Mainframes

Possible settings	any character string	See below.
Default setting	HELLO	
Dynamic specification	YES	
Specification within session	NO	

The stack can contain a sequence of Natural commands and/or user-specified commands, together with their data, for execution at the beginning of the Natural session.

The command stack is processed before the user is prompted for input on the screen (TP mode) or data are read from CMSYNIN/CMOBJIN files (batch mode).

If an INPUT statement is encountered during stack processing, the corresponding input screen is generated only if the required input data were not supplied with the command when the stack was created. Any reports generated during stack processing are displayed as usual.

Each system or user-defined command can be optionally followed by data which are used to satisfy requests for information required during the processing of the command. The character string provided as data for the STACK parameter must be enclosed in parentheses. If the command is a user command (that is, the name of a user program), any data provided resolve the data requirements of INPUT statements within the user program.

Conventions:

- Multiple settings for one INPUT statement are separated by a comma (,).
- Data for multiple INPUT statements are separated by a colon (:).
- Commands are separated by a semicolon (;).

Examples:

```
STACK=(LOGON USER1;UCMD1 A,B;UCMD2 C,D:E;FIN)

STACK=OFF                          No STACK data.

STACK=UCMND Execute command UCMND  No embedded blanks).

STACK=(CMD DATA:DATA;CMD...)       Place commands/data on stack.
```

Since some commands (for example, GLOBALS) do not read parameters by INPUT, a blank character should be used rather than a colon to delimit a command from the first parameter data element.

```
STACK='LOGON SYSTEM'
```

Because the macro assembler does not allow embedded blanks within parentheses, the character string must be enclosed in apostrophes when specified as static parameter.

UNIX/OpenVMS and Windows

Possible settings	any character string	
Default setting	none	
Dynamic specification	YES	
Specification within session	NO	

This parameter is used to place data/commands on the Natural command stack. The amount of data to be passed with this parameter is limited to 512 bytes. If this limit is exceeded, a corresponding error message is returned.

The stack can contain a sequence of Natural commands and/or user-specified commands, together with their data, for execution at the beginning of the Natural session. The command stack is processed before the user is prompted for input on the screen.

If an INPUT statement is encountered during stack processing, the corresponding input screen is generated only if the required input data were not supplied with the command when the stack was created. Any reports generated during stack processing are displayed as usual.

Each system or user-defined command can be optionally followed by data which are used to satisfy requests for information required during the processing of the command. If the command is a user command (that is the name of a user program), any data provided resolve the data requirements of INPUT statements within the user program.

Multiple settings for one INPUT statement are separated by a comma. Data for multiple INPUT statements are separated by a colon (:). A semicolon (;) is used to delimit multiple commands.

Examples:

LOGON:USER1;UCMD1 A,B;UCMD2 C,D:E;FIN

Logs on to the library USER1, executes the commands UCMD1 and UCMD2 providing the corresponding input data, and ends the Natural session.

CMD DATA:DATA;CMD

Places commands and data on stack. Since some commands (for example, GLOBALS) do not read parameters by INPUT, a blank character should be used rather than a colon to delimit a command from the first parameter data element.

If specified dynamically, the character string provided as data for the STACK parameter must be enclosed in apostrophes; if the character string contains multiple commands, it must also be enclosed in parentheses.

Example:

STACK="(logon system)"

 Copyright © Software AG 2002

Logs on to the library SYSTEM and executes the command UCMND.

STARTUP - Program Name for System Variable *STARTUP

This Natural profile parameter is for UNIX/OpenVMS and Windows platforms only.

It specifies a program name for the Natural system variable "*STARTUP ". The program whose name is contained in "*STARTUP" is executed each time the NATURAL command line is invoked. In a Natural program, you can assign another program name to "*STARTUP".

Possible settings	1 - 8 characters	Valid program name.
Default setting	none	
Dynamic specification	NO	
Specification within session	NO	

Note:
If you have Natural Security installed, STARTUP is not evaluated; the startup program to be used is read from the library profile defined in Natural Security.

 Copyright © Software AG 2002

STEPLIB - Default Steplib Library

This Natural profile parameter is for:

- mainframes
- UNIX/OpenVMS and Windows

Mainframes

This parameter determines the name of the default Natural steplib (concatenated library) to be used.

Possible settings	1 to 8 characters	Steplib name.
Default setting	SYSTEM	
Dynamic specification	YES	
Specification within session	NO	

UNIX/OpenVMS and Windows

This is a set of parameters that can only be specified with the help of the Natural Configuration Utility. It names up to eight libraries that can be searched for objects which cannot be found in the current library. The specification includes DBID and FNR of the system file where the library is located.

Possible settings	see text	The names of up to 8 libraries.
Default setting	NONE	
Dynamic specification	NO	
Specification within session	NO	

Search for Objects to be Executed from User Libraries

- current library (as defined by the system variable *LIBRARY-ID),
- steplibs (in sequence as specified in the Natural Security profile of the current library or in the **steplib** table),
- default steplib (as defined by the system variable *STEPLIB),
- library SYSTEM on the FUSER system file,
- library SYSTEM on the FNAT system file ([2] new!).

Search for Objects to be Executed from System Libraries

- current "SYS" library (as defined by the system variable *LIBRARY-ID),
- steplibs (in sequence as specified in the Natural Security profile of the current library or in **steplib** table),
- default steplib (as defined by the system variable *STEPLIB),
- library SYSTEM on the FNAT system file,
- library SYSTEM on the FUSER system file.

Thus, it is possible to make user exits generally available without having to keep copies of them on both system files. It will be sufficient to provide them in one location, namely on the FUSER system file.

SUBSID - Subsystem ID under OS/390 and VSE/ESA

This Natural profile parameter is for mainframes and OS/390 and VSE/ESA only.

It identifies the Natural subsystem to be used.

Possible settings	1 to 4 characters	Natural subsystem. If you specify less than 4 characters, blanks will be appended to it so as to get a 4-byte setting.
Default setting	NAT3	
Dynamic specification	YES	
Specification within session	NO	

For information on the Natural subsystem, see Natural Subsystem under OS/390 or Natural Subsystem under VSE/ESA.

 Copyright © Software AG 2002

SYMGEN - Generate Symbol Table

This Natural profile parameter is for UNIX/OpenVMS and Windows platforms only.

It determines whether a symbol table is generated.

Possible settings	ON	A symbol table will be generated.
	OFF	No symbol table will be generated.
Default setting	OFF	
Dynamic specification	YES	
Specification within session	YES	

The symbol table contains all symbols used within a Natural program (for example, variable names). It is part of the generated program and is required, for example, for the Natural Debugger and the dialog editor. See also session parameter SYMGEN.

SYNERR - Control of Syntax Errors

This Natural profile parameter is for:

- mainframes
- UNIX/OpenVMS and Windows

It determines whether or not syntax errors will be passed to the error transaction program.

Possible settings	ON	Syntax errors are passed to the error transaction program.
	OFF	Syntax errors are not passed to the error transaction program.
Default setting	OFF	
Dynamic specification	YES	
Specification within session	NO	

The error transaction program is defined either with the ETA profile parameter or within the Natural Security library profile.

 Copyright © Software AG 2002

SYS - Activate Set of Dynamic Profile Parameters

This Natural profile parameter is for mainframes only.

With this parameter you can activate a set of dynamic profile parameters.

In the Natural parameter module (NATPARM), you can use NTSYS macros to define sets of dynamic profile parameters. You identify such a set of parameters by giving it a unique set name.

Possible settings	1 to 8 characters	Set-name. For details, see NTSYS macro.
Default setting	none	
Dynamic specification	YES	This parameter can only be specified dynamically.
Specification within session	NO	

To activate a set of dynamic profile parameters, you specify its set name with the SYS parameter.

Example for OS/390:

```
// EXEC PGM=NATDEMO,PARM='SYS=SET2',...
```

To restrict the use of a set of dynamic profile parameters, you can use the USER parameter.

SYSCIP - Adabas Cipher Key for Natural System Files

This Natural profile parameter is for mainframes and only applies to Adabas databases.

It provides a default Adabas cipher key for access to Natural system files (FNAT, FUSER, FDIC, FSEC, FSPOOL) which have been loaded with the ciphered option.

Possible settings	8 characters, or blanks	The cipher code specified with the SYSCIP parameter applies to all Natural system files for which no individual cipher codes are specified. If the Natural system files are not ciphered, set SYSCIP to blanks.
Default setting	blanks	
Dynamic specification	YES	
Specification within session	NO	

Cipher codes for individual system files can be specified with the parameters FNAT, FUSER, FDIC, FSEC and FSPOOL.

If you specify the SYSCIP parameter dynamically in conjunction with any of the individual system file parameters FNAT, FUSER, FDIC, FSEC and FSPOOL, you must specify the SYSCIP parameter **before** any individual system file parameter.

 Copyright © Software AG 2002

SYSPSW - Adabas Password for Natural System Files

This Natural profile parameter is for mainframes and only applies to Adabas databases.

It provides a default Adabas password for access to Natural system files (FNAT, FUSER, FDIC, FSEC, FSPOOL) which have been password-protected.

Possible settings	8 characters, or blanks	If a Natural system file is password-protected, a password which permits updates to the file must be specified. The password specified with the SYSPSW parameter applies to all Natural system files for which no individual passwords are specified. If the Natural system files are not password-protected, set SYSPSW to blanks.
Default setting	blanks	
Dynamic specification	YES	
Specification within session	NO	

Passwords for individual system files can be specified with the profile parameters FNAT, FUSER, FDIC, FSEC and FSPOOL.

If you specify SYSPSW dynamically in conjunction with any of the individual system file parameters FNAT, FUSER, FDIC, FSEC and FSPOOL, you must specify SYSPSW **before** any individual system file parameter.

If the OPRB parameter is specified, the SYSPSW password is used for the initial Adabas open call and must permit access and/or update to all the files specified in OPRB as required.

TAB - Standard Output Translation Table

This Natural profile parameter is for mainframes only.

It can be used to specify dynamically the same options which you can specify statically with the NTTAB macro in the parameter module.

Possible settings	See NTTAB Syntax.	See NTTAB macro.
Default setting	none	
Dynamic specification	YES	This parameter can only be specified dynamically.
Specification within session	NO	

With the TAB parameter, you must enclose the entire string of character pairs in parentheses, for example:

```
TAB=(5E,'Ä','ö',78,FF,00,'ü','Ü')
```

 Copyright © Software AG 2002

TAB1 - Alternative Output Translation Table

This Natural profile parameter is for mainframes only.

It can be used to specify dynamically the same options which you can specify statically with the NTTAB1 macro in the parameter module.

Possible settings		See NTTAB1 macro.
Default setting	none	
Dynamic specification	YES	This parameter can only be specified dynamically.
Specification within session	NO	

TAB2 - Alternative Input Translation Table

This Natural profile parameter is for mainframes only.

It can be used to specify dynamically the same options which you can specify statically with the NTTAB2 macro in the parameter module.

Possible settings		See NTTAB2 macro.
Default setting	none	
Dynamic specification	YES	This parameter can only be specified dynamically.
Specification within session	NO	

 Copyright © Software AG 2002

TABA1 - EBCDIC-to-ASCII Translation Table

This Natural profile parameter is for mainframes only.

It can be used to specify dynamically the same options which you can specify statically with the NTTABA1 macro in the parameter module.

Possible settings		See NTTABA1 macro.
Default setting	none	
Dynamic specification	YES	This parameter can only be specified dynamically.
Specification within session	NO	

TABA2 - ASCII-to-EBCDIC Translation Table

This Natural profile parameter is for mainframes only.

It can be used to specify dynamically the same options which you can specify statically with the NTTABA2 macro in the parameter module.

Possible settings		See NTTABA2 macro.
Default setting	none	
Dynamic specification	YES	This parameter can only be specified dynamically.
Specification within session	NO	

 Copyright © Software AG 2002

TABL - Translation Table for Output from "SYS" Libraries

This Natural profile parameter is for mainframes only.

It can be used to specify dynamically the same options which you can specify statically with the NTTABL macro in the parameter module.

Possible settings		See NTTABL macro.
Default setting	none	
Dynamic specification	YES	This parameter can only be specified dynamically.
Specification within session	NO	

TD - Time Differential

This Natural profile parameter is for:

- mainframes
- UNIX/OpenVMS and Windows

It indicates a time differential to be applied to the Natural time/date setting to ensure that the current local time/date is used, rather than the computer center time/date.This parameter is applicable in an environment in which remote nodes are being used in a computer network.

Mainframes

Possible settings	AUTO	Natural compares the physical (store clock) and logical (system environment) machine times and uses the difference between the two as the setting for the TD parameter. For a time change to take effect for Natural (for example, to change time to summer time or back to winter time), it is therefore sufficient to reset the logical machine time.
	or -23,59 to 23,59	Hours and minutes. If TD=*hours* or TD=(*hours*,*minutes*) is specified, the specified time is added to / subtracted from the physical machine time to set the time/date to be used by Natural.
Default setting	0	
Dynamic specification	YES	
Specification within session	NO	

Examples:

```
TD=6           (6 hours ahead)
TD=(5,30)      (5 hours and 30 minutes ahead)
TD=(-6,12)     (6 hours and 12 minutes behind)
```

VSE/ESA-specific information:

With VSE-type operating systems, "// ZONE" and "//DATE" JCL statements are honored with TD=AUTO. This can also affect the setting of the profile parameter DD. See also the profile parameters YD and DD.

UNIX/OpenVMS and Windows

 Copyright © Software AG 2002

Possible settings	-23,59 to 23,59	The notation "*hours*,*minutes*" is used to add/subtract the specified time to/from the physical machine time to set the time/date to be used by Natural; "*minutes*" (if specified) must be "00" to "59".
Default setting	0,0	
Dynamic specification	YES	
Specification within session	NO	

Examples:

```
TD=6        (6 hours ahead)
TD=-11      (11 hours behind)
TD=(5,30)   (5 hours and 30 minutes ahead)
TD=(-6,30)  (6 hours and 30 minutes behind)
```

TF - Translation of Database ID/File Number

This Natural profile parameter is for:

- mainframes
- UNIX/OpenVMS and Windows

The parameter TF applies only to user files, **not** to system files.

This parameter can be used to translate a database ID/file number to another database ID/file number.

The translation of file number is relevant when developing an application in a production environment; it enables you to develop an application in a test database and then transfer the finished application to the production database without having to change or re-compile the application.

The transfer is done by a translation of the application's database identification (DBID) and file number (FNR): with the TF parameter, you specify the production DBID/FNR and the test DBID/FNR. The Natural objects are then cataloged with the production DBID/FNR, but whenever a database access is executed, the production DBID/FNR is translated into the test DBID/FNR; that is, the test database is used. This means that testing can take place in the actual production environment, but not with production data.

Mainframes

This parameter can be used to dynamically specify the same options which can be specified statically with the macro NTTF in the parameter module.

Possible settings	production and test database IDs, and production and test file numbers	See below.
Default setting	none	
Dynamic specification	YES	This parameter can only be specified dynamically.
Specification within session	NO	

The parameter is specified as follows:

`TF=(`*`production-DBID`*`,`*`production-FNR`*`,`*`test-DBID`*`,`*`test-FNR`*`)`

To translate all FNRs in the specified *production-DBID* to all FNRs in the specified *test-DBID*, you specify an asterisk for both *production-FNR* and *test-FNR*:
`TF=(`*`production-DBID`*`,*,`*`test-DBID`*`,*)`

To translate *all* production DBIDs and FNRs to the specified *test-DBID* and *test-FNR*, you specify asterisks for the *production-DBID* and *production-FNR*:
`TF=(*,*,`*`test-DBID`*`,`*`test-FNR`*` )`

The TF parameter can be specified several times so as to specify different combinations of file numbers.

 Copyright © Software AG 2002

Instead of the TF parameter, you can also use the NTTF macro in the parameter module. To transfer the finished application from testing to production, you simply switch off the DBID/FNR translation by using another Natural nucleus in which the NTTF macro is omitted from the NATPARM module: the programs then use the original DBID/FNR with which they were cataloged; that is, the production DBID/FNR.

UNIX/OpenVMS and Windows

This parameter is used to translate the database ID/file number of a production database into the database ID/file number of a test database.

Possible settings	see text "*" for production DBID and FNR
Default setting	none
Dynamic specification	YES
Specification within session	NO

The TF parameter can be specified several times. Existing specifications are displayed in a corresponding list box.

Note:
Production and test databases must be of the same type (Adabas/Adabas, for example).

Note:
For SQL databases, the file number must always be set to 1.

TIMEOUT - Wait Time for RPC Server Response

This Natural profile parameter is for UNIX/OpenVMS and Windows platforms only.

Note for mainframes:
Use the keyword parameter TIMEOUT of the NTRPC macro for static specification, or the subparameter with the same name of the profile parameter RPC.

It specifies the number of seconds the client is to wait for an RPC server response. If this time is exceeded, the remote procedure call will be terminated with a corresponding error message.

TIMEOUT is specified on the client side only.

Possible settings	0 - 32767	Seconds.
Default setting	55	
Dynamic specification	YES	
Specification within session	YES	At runtime, this value can be overwritten by the parameter maintenance of the SYSRPC utility.

For further information see the Natural Remote Procedure Call documentation.

 Copyright © Software AG 2002

TMPSORTUNIQ - Alternate Algorithm for Generating Sort Work File Names

This Natural profile parameter is for UNIX/OpenVMS and Windows platforms only.

If the parameter is specified on the Natural startup command, Natural will use a different algorithm from normal when generating work file names for sort operations. It forces Natural to generate a unique file name for any required work files without embedding the values of the ETID parameter and user ID into the file name.

The normal, default algorithm creates work file names with the values of the ETID setting and the user ID embedded. Whereas this algorithm usually results in file names which are unique to a particular Natural session, this may cause problems in environments where the ETID contains characters which are invalid within a file name, or where multiple Natural sessions are running which use the same user ID and no ETID specification (thus possibly resulting in work file names for sort operations being created which are not unique).

Possible settings	specified or not specified	If specified, Natural will use a different algorithm from normal when generating work file names for sort operations.
Default setting	not specified	
Dynamic specification	YES	
Specification within session	NO	

TPF (Internal Use)

This Natural profile parameter is for mainframes only.

This parameter is reserved for internal use by Natural. Do not change its setting.

 Copyright © Software AG 2002

TQ - Translate Quotation Marks

This Natural profile parameter is for:

- mainframes
- UNIX/OpenVMS and Windows

It controls the translation of a quotation mark (") within a Natural text constant. It takes effect at compilation time only.

Possible settings	ON	Each quotation mark within a text constant is output as a single apostrophe.
	OFF	Quotation marks within text constants are not translated, they are output as quotation marks.
Default setting	ON	
Dynamic specification	YES	
Specification within session	NO	

Note:
Do not confuse quotation mark (") with double apostrophes (’’). Double apostrophes within a text constant are always output as a single apostrophe (’), regardless of the setting of the TQ parameter.

Example 1 (TQ=ON):

```
WRITE ’THERE’’S A QUOTATION MARK’
```

is displayed as: THERE’S A QUOTATION MARK

Example 2 (TQ=OFF):

```
WRITE ’THERE"S A QUOTATION MARK’
```

is displayed as: THERE"S A QUOTATION MARK

TRACE - Define Trace Level for Natural RPC Servers

This Natural profile parameter is for UNIX/OpenVMS and Windows.

Note for mainframes:
Use the keyword parameter TRACE of the NTRPC macro for static specification, or the subparameter with the same name of the profile parameter RPC.

It activates the RPC trace facility and determines the trace level *n* to be used. For further information, see Using the Server Trace Facility p.p. in the Natural Remote Procedure Call documentation.

Possible settings	0	Nothing is traced.
	1	Only messages (inclusive Natural errors) are traced.
	2	All messages and data from/to client are traced.
Default setting	0	
Dynamic specification	YES	
Specification within session	NO	

The values 3-9 displayed in the selection box are for future use. If one of these values is chosen, then TRACE=2 is selected.

 Copyright © Software AG 2002

TRACE - Define Mainframe Components to be Traced

This Natural profile parameter is for mainframes only.

Do not use this parameter without prior consultation of Software AG Support.

The TRACE parameter corresponds to the macro NTTRACE in the parameter module. It can be used to specify dynamically the same options which you can specify statically with the macro NTTRACE.

Possible settings	List of character strings (each 1-8 bytes)	List of trace IDs. As trace-IDs, you specify the components to be traced.
Default setting	none	
Dynamic specification	YES	This parameter can only be specified dynamically.
Specification within session	NO	

The setting lists of multiple TRACE parameters are not concatenated; that is, a TRACE parameter overrides any previously specified TRACE parameters or NTTRACE macros.

For further information see the Natural Remote Procedure Call documentation.

TRANSP - Server Transport Protocol

This Natural profile parameter is for UNIX/OpenVMS and Windows platforms only.

Note for mainframes:
Use the keyword parameter TRANSP of the NTRPC macro for static specification, or the subparameter with the same name of the profile parameter RPC.

It determines which server transport protocol is used. If ACI is used, you can additionally specify the transport method.

TRANSP is specified on the server side only.

Possible settings	ACI	ACI is used. The transport method is defined by the EntireX Broker.
	CSCI	CSCI is used. (OpenVMS only).
	ACI,TCP	ACI is used with TCP/IP.
	ACI,NET	ACI is used with Entire Net-work, i.e. using the Adabas protocol.
	ACI,TCP-NET	Trying to use ACI with TCP. If not available, ACI is used with NET.
	ACI,NET-TCP	Trying to use ACI with NET. If not available, ACI is used with TCP.
Default setting	ACI	
Dynamic specification	YES	
Specification within session	NO	

For further information see the Natural Remote Procedure Call documentation.

 Copyright © Software AG 2002

TRYALT - Try Alternative Server Address

This Natural profile parameter is for UNIX/OpenVMS and Windows platforms only.

Note for mainframes:
Use the keyword parameter TRYALT of the NTRPC macro for static specification, or the subparameter with the same name of the profile parameter RPC.

It determines whether an RPC client should try to execute an RPC request on an alternative server (ON) or not (OFF). For further information, see Specifying RPC Server Adresses in the Natural Remote Procedure Call documentation.

TRYALT is specified on the client side only.

Possible settings	ON	If a request could not be executed on the node you specified, the RPC client tries to find an alternative server address to send that request to.
	OFF	No such attempt will be made.
Default setting	OFF	
Dynamic specification	YES	
Specification within session	YES	At runtime, this value can be overwritten by the parameter maintenance of the SYSRPC utility.

For further information see the Natural Remote Procedure Call documentation.

TS - Translate Output from Programs in System Libraries

This Natural profile parameter is for mainframes only.

It is used to translate output from Natural system libraries (that is, libraries whose names begin with "SYS") using a translation table. This may be necessary for locations which have non-standard lower-case usage (for example, Middle East countries); see also the profile parameter PM.

Possible settings	ON	Output is translated.
	OFF	Output is not translated.
Default setting	OFF	
Dynamic specification	YES	
Specification within session	YES	Within a Natural session, the profile parameter TS can be overridden by the **session parameter** TS.

In addition, with TS=ON, the profile parameter LC=OFF and the session parameter AD=T, both of which translate input to upper case, are ignored, as they would cause undesired character translation for special character sets.

The translation table can be modified with the NTTABL macro or the corresponding dynamic profile parameter TABL.

The TS parameter applies only to primary output (CMPRINT).

 Copyright © Software AG 2002

TSIZE - Size of Buffer for Adabas Text Retrieval

This Natural profile parameter is for mainframes only.

It determines the size of the buffer to be used for the Adabas Text Retrieval facility. If the requested space is not available, the Adabas Text Retrieval facility cannot be used.

Possible settings	0 - 128	Buffer size in KB.
Default setting	0	Adabas Text Retrieval facility is not used.
Dynamic specification	YES	
Specification within session	NO	

TTYPE - Terminal Type

This Natural profile parameter is for mainframes only.

It allows you to specify the terminal type used - in TP environments in which this information is not supplied automatically - so that Natural can activate the appropriate converter routine for attribute sequences to operate that type of terminal.

Possible settings	1-4 characters	The setting specified with the TTYPE parameter must be defined as a valid terminal device type in the NTDVCE macro of the NATCONFG module, see Configuration Tables - Module NATCONFG.
Default setting	IBM	3270
	Siemens	The setting defined in PDN, unless overridden by the parameter T975X (see Installing the Natural TIAM Interface in the Natural for Mainframes Installation Guide).
Dynamic specification	YES	
Specification within session	YES	The TTYPE parameter has the same function as the terminal command %T= (described in the Natural Reference documentation).

If you use the TTYPE parameter, it is no longer necessary to execute a program containing a SET CONTROL ’T=...’ statement at the start of the session in order to set the terminal type.

 Copyright © Software AG 2002

UDB - User Database ID

This Natural profile parameter is for:

- mainframes
- UNIX/OpenVMS and Windows

It specifies the DBID to be used for a database access.

Mainframes

Possible settings	0 - 65535 (except 255)	Valid database ID.
Default setting	database ID applicable for FUSER	
Dynamic specification	YES	
Specification within session	NO	

If no DBID is specified in the DDM used, the DBID specified with the UDB parameter determines which database is accessed. Thus it is possible to have different user environments without multiple FUSER files being required.

If no DBID is specified in the DDM and the UDB parameter is not specified, the DBID that applies to the FUSER system file is used.

UNIX/OpenVMS and Windows

Possible settings	0 - 65535 (except 255)	Valid database ID.
Default setting	1	
Dynamic specification	YES	
Specification within session	NO	

If no DBID is specified in the DDM used, the DBID specified with the UDB parameter determines which database is accessed. If so, the UDB parameter must be set to a valid DB number.

UDB is also used to specify the DBID for stowing transaction data if the parameter ETDB is not specified.

ULANG - User Language

This Natural profile parameter is for:

- mainframes
- UNIX/OpenVMS and Windows

It indicates the language to be used for date edit masks, system messages, user messages, helptexts, helproutines, and multi-lingual maps. The setting is used to set the Natural system variable "*LANGUAGE".

Possible settings	1 - 60	Language code. For example, 1 is assigned to English, 2 is assigned to German, 3 is assigned to French. For a detailed list of language codes, see the table in section *LANGUAGE.
Default setting	1	
Dynamic specification	YES	
Specification within session	NO	

Mainframes:
For additional information about language indicators and possible settings, see Configuration Tables - Module NATCONFG.

 Copyright © Software AG 2002

USEDIC - Common Logical Name for Dictionary Servers

This Natural profile parameter is for UNIX/OpenVMS and Windows platforms only.

It is used to specify a common logical name for dictionary servers defined with Natural RPC to enable remote dictionary access on a mainframe or UNIX host. See also Dictionary Server Assignments.

Possible settings	any valid server name	Remote dictionary access will be possible.
Default setting	blank	Remote dictionary access will not be possible.
Dynamic specification	NO	
Specification within session	NO	

USER - Restrict the Use of Profile Parameter Strings

This Natural profile parameter is for:

- mainframes
- UNIX/OpenVMS and Windows

Mainframes

This parameter can be used to restrict the use of parameter strings as specified in a SYSPARM profile, NTSYS Macro or Dynamic Parameter Dataset (CMPRMIN).

Possible settings	list of user IDs	The IDs of the users who will be allowed to use the subsequently specified string of profile parameters. Only the specified users will then be allowed to use that parameter string.
Default setting	none	
Dynamic specification	YES	This parameter can only be specified dynamically.
Specification within session	NO	

The USER parameter applies only to the string of parameters specified **after** it.

You specify the USER parameter as follows:
USER=(*user-id*,...),*string of restricted profile parameters*

When the dynamic profile parameters are evaluated, Natural checks if the current user ID (that is, the current setting of the system variable *INIT-USER) is contained in the list of user IDs specified with the USER parameter. If it is not, the user receives a corresponding error message (NAT7017), and the processing of dynamic profile parameter is terminated immediately.

To restrict the use of:

- a SYSPARM profile to specific users,
  you specify the USER parameter as the first parameter in the profile. The subsequent string of profile parameters in the profile, that is, the entire profile, can then only be used by the user specified with the USER parameter.
- an NTSYS macro,
  you specify the USER parameter as the first parameter in the string of parameters after the set name of the dynamic profile parameters. In the same way you can restrict the use of profile parameter strings in a CMPRMIN dataset.
- an alternative parameter module (as specified with the PARM parameter),
  you can use the NTUSER Macro.

UNIX/OpenVMS and Windows

This parameter is used to enter a user ID. If the default setting is used, the login user ID from the operating system applies.

 Copyright © Software AG 2002

Possible settings	any valid user ID	
Default setting	blank	
Dynamic specification	NO	
Specification within session	NO	

USERBUF (Internal Use)

This Natural profile parameter is for mainframes only.

This parameter is reserved for internal use by Natural. Do not change its setting.

 Copyright © Software AG 2002

USEREP - Repository Usage

This Natural profile parameter is for Windows platforms only.

It enables you to use the repository.

Possible settings	ON	Repository usage enabled.
	OFF	Repository usage disabled.
Default setting	ON	
Dynamic specification	YES	
Specification within session	NO	

USIZE - Size of User Buffer

This Natural profile parameter is for UNIX/OpenVMS and Windows platforms only.

It indicates the size of the user buffer in virtual memory. The user buffer contains all data dynamically allocated by Natural.

Possible settings	0 or 10 - 1024	Buffer size in MB.
Default setting	10	
Dynamic specification	NO	
Specification within session	NO	

Note:
If USIZE is set to zero, the memory capacity will be unrestricted.

 Copyright © Software AG 2002

UTAB1 - Translation Table for Lower to Upper Case

This Natural profile parameter is for mainframes only.

It can be used to specify dynamically the same options which you can specify statically with the NTUTAB1 macro in the parameter module.

Possible settings		See NTUTAB1 macro.
Default setting	none	
Dynamic specification	YES	This parameter can only be specified dynamically.
Specification within session	NO	

UTAB2 - Translation Table for Upper to Lower Case

This Natural profile parameter is for mainframes only.

It can be used to specify dynamically the same options which you can specify statically with the NTUTAB2 macro in the parameter module.

Possible settings		See NTUTAB2 macro.
Default setting	none	
Dynamic specification	YES	This parameter can only be specified dynamically.
Specification within session	NO	

 Copyright © Software AG 2002

V22COMP - Compatibility with Inconsistent Version 2.2 Syntax

This Natural profile parameter is for UNIX/OpenVMS and Windows platforms only.

It provides a compatibility option for inconsistent Natural Version 2.2 syntax. Use this option for a smooth transition from Version 2.2 to Version 3.1 or above.

Note:
The V22COMP option will be available only for a limited period of time to allow a smooth transition to the syntax of Version 3.1 or above. It will be removed again with a subsequent release of Natural.

Possible settings	ON	The syntax constructions described below will *not* lead to a syntax error. Thus you are able to compile your existing programs under Version 3.1 or above until you have adjusted them to the requirements of Version 3.1 or above.
	OFF	These syntax constructions will lead to a syntax error.
Default setting	OFF	
Dynamic specification	NO	
Specification within session	NO	

The following inconsistent syntax constructions that were not intercepted by Version 2.2 lead to a syntax error with Version 3.1 or above:

DEFINE DATA: inconsistent number of decimal digits in constant setting;
DEFINE WINDOW: specified window size smaller than minimum.

VSIZE - Size of Buffer Area for Natural/VSAM

This Natural profile parameter is for mainframes and applies to Natural for VSAM only.

It sets the maximum size of the buffer area required by Natural for VSAM. If set to 0 or if the requested space is not available, the Natural for VSAM Interface cannot be used.

Possible settings	0 - 512	Buffer size in KB.
Default setting	0	
Dynamic specification	YES	
Specification within session	NO	

The size actually required depends on the specifications in the NVSPARM macro (described in the Natural for VSAM documentation).

If VSIZE is set to 0 or if the size specified with the VSIZE parameter is not sufficient, an appropriate error message at the initialization of Natural for VSAM tells you which buffer specified in NVSPARM does not fit into the VSIZE area; you can then either reduce individual buffer sizes in NVSPARM or increase the size of the VSIZE area.

If Natural is installed for VSAM, the corresponding Natural buffers are requested at the initialization of the Natural session. If you do not need VSAM support during a Natural session, it is recommended that you invoke Natural with VSIZE=0 to avoid overhead caused by handling of unused buffers.

 Copyright © Software AG 2002

WFOPFA - Opening of Work Files

This Natural profile parameter is for UNIX/OpenVMS and Windows platforms only.

It specifies when work files are to be opened by Natural.

Possible settings	ON	A work file is opened at the time when it is first accessed by a given READ WORK FILE or WRITE WORK FILE statement. This means that only those work files which are actually accessed are opened, while the contents of unopened work files are not deleted.
	OFF	All work files referenced in a Natural object are opened when the execution of this object starts. (This may delete the content of a work file when closing if the work file was referenced by a WRITE WORK FILE statement that was never executed.)
Default setting	ON	
Dynamic specification	YES	
Specification within session	NO	

WFOPFA=OFF only affects main programs; for routines, WFOPFA=ON always applies.

WH - Wait for Record in Hold Status

This Natural profile parameter is for:

- mainframes
- UNIX/OpenVMS and Windows

It specifies the action to be taken if a required record is not available for processing.

Possible settings	ON	The user is placed in wait status until either the requested record becomes available or an error message is issued due to Adabas exceeding a time limit or other limit while attempting to place the record in hold status.
	OFF	An error message is returned if any of these records cannot be placed in hold status.
Default setting	OFF	
Dynamic specification	YES	
Specification within session	YES	Within a Natural session, the profile parameter WH can be overridden by the **session parameter** WH.

When a Natural statement is executed which results in Adabas records being read and an update/delete operation could follow, Natural requests that Adabas places these records in hold status. See the Adabas Command Reference documentation for further information on hold processing.

 Copyright © Software AG 2002

WORK - Work-File Assignments

This Natural profile parameter is for:

- mainframes
- UNIX/OpenVMS and Windows

Mainframes

It corresponds to the NTWORK macro in the parameter module. It can be used to specify dynamically the same options which you can specify statically with the NTWORK macro.

Possible settings		see NTWORK macro
Default setting		
Dynamic specification	YES	This parameter can only be specified dynamically.
Specification within session	NO	

See also FAMSTD - Overwriting of Print and Work File Access Method Assignments.

UNIX/OpenVMS and Windows

This parameter indicates the number of work files available.

Possible settings	0 - 32	Number of work files.
Default setting	0	
Dynamic specification	NO	
Specification within session	NO	

WORK=*nn*

If WORK is set to "*nn*" (in the range 1 - 32), this setting defines the highest work file number which is available for the Natural session. If a work file has been previously defined with a higher number, this new setting overrides the previous higher setting.

WPSIZE - Sizes of Natural Work Pools

This Natural profile parameter is for mainframes only.

It determines the sizes of the Natural work pools below and above the 16 MB line. In these work pools, all temporary work buffers for the Natural nucleus are allocated.

Possible settings	(0,0) to (1024,16384)	Work pool sizes in KB. Specification see below.
Default setting	(32,128)	
Dynamic specification	YES	
Specification within session	NO	

The WPSIZE parameter is specified in the form **WPSIZE=(*n*,*m*)**.
n defines the size of the work pool *below* the 16 MB line.
m defines the size of the work pool *above* the 16 MB line.

Natural allocates the work pools outside the Natural storage thread according to the specified settings.

If "0" is specified for *n* and/or *m*, no dynamic work pool will be allocated below and/or above 16 MB, and all physical GETMAIN requests will be passed to the operating system.

For non-thread environments (e.g. batch, TSO) the recommended setting is WPSIZE=(0,0). This may save buffer storage.

 Copyright © Software AG 2002

WSISIZE - Buffer for Natural Workstation Interface

This Natural profile parameter is for mainframes and only applies if Natural Workstation Interface is installed.

It determines the size of the buffer area used by Natural Workstation Interface. See the Natural Workstation Interface documentation.

Possible settings	0, or 10 - 256	Size of buffer area in KB.
Default setting	0	
Dynamic specification	YES	
Specification within session	NO	

If WSISIZE=0 or if the required space is not available, the Natural Workstation Interface cannot be used.

WSIZE - Size of Buffer Area for Screen Images

This Natural profile parameter is for mainframes only.

It determines the maximum size of the buffer area where the screen images are stored that are saved when the **terminal command** %WA has been entered. The buffer area will be created only **after** %WA has been entered.

Possible settings	1 - 64	Size of buffer area in KB.
Default setting	12	
Dynamic specification	YES	
Specification within session	NO	

If WSIZE=0 or if the required space is not available, the screen images cannot be saved with the terminal command "%WA".

 Copyright © Software AG 2002

XADB - Database for Coordination of Transaction Processing

This Natural profile parameter is for UNIX/OpenVMS and Windows platforms only. It is reserved for future use.

It specifies the databases, where transaction processing is to be coordinated by a TP-monitor system. Mark all such databases in the parameter's database list box.

Possible settings	ON / OFF	Use the check boxes to specify the relevant databases.
Default setting	none	
Dynamic specification	NO	
Specification within session	NO	

XREF - Active Cross-Reference Feature

This Natural profile parameter is for:

- mainframes
- UNIX/OpenVMS and Windows

Mainframes

This parameter indicates whether the active cross-reference feature is to be used.

Possible settings	ON	Natural cross-reference data for database fields, subroutines and maps are stored in the appropriate Predict entries each time a Natural program is cataloged.
	OFF	No cross-reference activity is performed.
	FORCE	A check is made to determine if a Predict entry for the program exists: if so, CATALOG is allowed and cross-reference data are stored; if not, CATALOG is disallowed.
	DOC	Same as XREF=FORCE, but without generating any cross-reference data.
Default setting	OFF	
Dynamic specification	YES	
Specification within session	NO	

UNIX/OpenVMS and Windows

This parameter indicates how to handle Natural cross-reference data.

Possible settings	ON	Natural cross-reference data for various Natural objects, database fields, subroutines and maps are stored in the appropriate repository each time a Natural object is cataloged (with the CATALL or STOW system command).
	OFF	No cross-reference data is generated. Existing Natural cross-reference data is deleted from the repository.
	FORCE	With the repository, option FORCE works as option ON. With Predict, XREF data are only stored if the Natural object is documented in Predict. If it is not, an error is returned.
	DOC	Same as XREF=FORCE, but without generating any cross-reference data.
Default setting	OFF	
Dynamic specification	YES	
Specification within session	NO	

 Copyright © Software AG 2002

XSIZE - Size of Buffer for User Subsystem

This Natural profile parameter is for mainframes only.

It specifies the size of the buffer area to be used for user subsystems called by Natural programs.

If Natural Connection is installed and asynchronous lines are used, the XSIZE parameter is reserved for internal use by Natural Connection and must not be used otherwise.

Possible settings	0 - 64	Size of the buffer area in KB.
Default setting	0	If XSIZE=0 or if the required space is not available, the user subsystem cannot be used.
Dynamic specification	YES	
Specification within session	NO	

YD - Year Differential

This Natural profile parameter is for mainframes only.

It can be used to adjust the current machine date (as read by using the internal machine time) by adding/subtracting a number of years to/from it. This may be useful for countries that use different calendars.

Possible settings	-499 to 499	The parameter is specified as "**YD=+***nnn*" or "**YD=-***nnn*" (*nnn* being the number of years).
Default setting	0	
Dynamic specification	YES	
Specification within session	NO	

If the current year is a leap year, but the year resulting from the YD setting is not, the 1st March will be used instead of the 29th February.

The year resulting from the sum of the profile parameters TD, DD and YD must be in the range of 1583 through 2699.

 Copyright © Software AG 2002

YSLW - Year Sliding Window

This Natural profile parameter is for:

- mainframes
- UNIX/OpenVMS and Windows

It determines the range of years covered by the "year sliding window".

Possible settings	0 - 99	Range of years. The sliding-window mechanism assumes a date with a 2-digit year to be within a "window" of 100 years. Within these 100 years, every 2-digit year setting can be uniquely related to a specific century. You determine how many years in the past that 100-year range is to begin: The YSLW setting is subtracted from the current year to determine the first year of the window range. For example, if the current year is 1999 and you specify YSLW=40, the sliding window will cover the years 1959 to 2058. A 2-digit year setting *nn* from 59 to 99 is then interpreted accordingly as 19*nn*, while a 2-digit year setting *nn* from 00 to 58 is interpreted as 20*nn*.
Default setting	0	No sliding-window mechanism will be used.
Dynamic specification	YES	
Specification within session	NO	

The YSLW parameter is evaluated at runtime when an alphanumeric date setting with a 2-digit year component is moved into a date variable. This applies to data settings which are: used with the mathematical function VAL, used with the IS(D) option in a logical condition, read from the stack as input data, or entered in a map as input data.

See also the section Processing of Date Information in the Natural Programming Guide documentation.

ZD - Zero-Division Check

This Natural profile parameter is for:

- mainframes
- UNIX/OpenVMS and Windows

It indicates the action to be taken when an attempt is made to perform a division operation in which the divisor is "0".

Possible settings	ON	Natural issues an error message if a division by "0" is attempted.
	OFF	Natural returns "0" for any division operation in which the divisor is "0".
Default setting	ON	
Dynamic specification	YES	
Specification within session	YES	Within a Natural session, the profile parameter ZD can be overridden by the **session parameter** ZD.

 Copyright © Software AG 2002

ZP - Zero Printing

This Natural profile parameter is for:

- mainframes
- UNIX/OpenVMS and Windows

It indicates how a field which contains a setting of all zeros is to be output.

Mainframes

Possible settings	ON	Each field setting which consists of all zeros is suppressed.
	OFF	Each field setting which consists of all zeros is output as one zero, right justified (for numeric fields) or all zeros (for time fields).
Default setting	ON	
Dynamic specification	YES	
Specification within session	YES	Within a Natural session, the profile parameter ZP can be overridden by the **session parameter** ZP.

ZP - Zero Printing for UNIX/OpenVMS and Windows

This parameter indicates how a field which contains a setting of all zeros is to be printed.

Possible settings	ON	Each field setting which consists of all zeros is output as one zero, right justified (for numeric fields) or all zeros (for time fields).
	OFF	Each field setting which consists of all zeros is suppressed.
Default setting	ON	
Dynamic specification	YES	
Specification within session	YES	Within a Natural session, the profile parameter ZP can be overridden by the **session parameter** ZP.

ZSIZE - Size of Entire DB Buffer Area

This Natural profile parameter is for mainframes and only applies to Entire DB.

It determines the size of the buffer area required by Entire DB.

Possible settings	0 - 64	Size of the buffer area in KB.
Default setting	0	If ZSIZE=0 or if the required space is not available, the Entire DB Interface cannot be used.
Dynamic specification	YES	
Specification within session	NO	

 Copyright © Software AG 2002

Session Parameters

Natural session parameters may be used with certain Natural statements to control such factors as the size of a report, how fields are to be displayed, etc.

This section covers the following topics:

- General Information
- How to Set Session Parameters
- Evaluation of Session Parameters
- Summary of Session Parameters

General Information

In Natural, several parameters are used:

- to specify certain characters,
- to set processing time limits,
- to set a particular response for a given condition,
- to set various size limits,
- to determine various aspects of output reports.

The statements in which a session parameter may be set/evaluated are indicated in the section Summary of Parameters as well as in the description of each parameter.

At the installation of Natural, the Natural administrator sets these parameters to default values which are then valid for all users of Natural.

To see which parameter values apply to your session, you enter the system command GLOBALS as described in the Natural User's Guide.

How to Set Parameters

Natural session parameters can be set in several ways:

- via the default parameter module/file (NATPARM), which is set when Natural is installed;
- via dynamic parameters specified when invoking Natural (as described in your Natural Operations documentation);
- via the system command GLOBALS;
- via a SET GLOBALS statement (in reporting mode only);
- via a FORMAT statement;
- via parameter specification within statements where parameters also are evaluated, for example, INPUT, DISPLAY, WRITE;
- via terminal commands.

Instead of the parameter values "ON" and "OFF", you can also specify "T" (true) or "F" (false) respectively.

Changing Parameters at Session Level - The GLOBALS Command

For your Natural session you can change some of the parameter values set by the Natural administrator.

Within your Natural session, you can change these parameters by issuing the following system command:

GLOBALS

When you issue the GLOBALS command, a screen is displayed showing the parameter values that are currently in effect for your session. On this screen, you can change the values that do not suit your requirements.

A parameter value set with a GLOBALS command remains in effect until the end of the Natural session (and applies to every object you store during the session), unless you change it again with another GLOBALS command.

Changing Parameters at Program Level - The FORMAT Statement

You can change certain parameters for the duration of a single program (report). This is done by using a FORMAT statement in the program, which will override the session-wide settings for these parameters.

Example of a FORMAT Statement:

```
FORMAT AL=10 HC=R
```

Parameters set with a FORMAT statement apply until the end of the executed program, unless they are changed with another FORMAT statement in the program.

Not all session parameters can be changed at program level, while several parameters that can be specified at program level cannot be specified at session level; most of the latter are parameters which affect the format of an output report.

Changing Parameters at Statement Level

Most of the parameters you can change with a FORMAT statement you can also change for an individual statement; for example, for a particular DISPLAY, WRITE, INPUT or REINPUT statement.

This is done by specifying the parameter (in parentheses) after the statement name.

Example:

```
DISPLAY (SF=4) NAME JOB-TITLE CURR-CODE SALARY
```

A parameter set at statement level applies only to the statement in which it is specified. The setting at statement level overrides, for that statement only, all other settings of that parameter at other levels.

 Copyright © Software AG 2002

Changing Parameters at Field Level

Within a DISPLAY, WRITE, INPUT or REINPUT statement, you can also change some parameters for an individual field or output element.

This is done by specifying the parameter (in parentheses) after the field name.

Example:

```
DISPLAY NAME (AL=10) JOB-TITLE CURR-CODE SALARY
```

The parameter value then applies only to that field. The setting at field level overrides, for that field only, all other settings of that parameter at other levels. However, only some of the parameters that can be set at statement level can also be set at field level.

Copyright © Software AG 2002

Evaluation of Parameters

Parameters specified with the statements DISPLAY, FORMAT, PRINT, INPUT, REINPUT, WRITE, WRITE TITLE and WRITE TRAILER are processed during program compilation and are therefore included in the corresponding object module for the program.

The following hierarchy is used for evaluation:

1. Parameters set at element/field (highest priority)
2. Parameters set at statement level
3. Parameters set with a FORMAT statement
4. The default parameter settings (lowest priority)

Parameters set with a SET GLOBALS statement cause the execution time environment to be modified. These modifications remain in effect until overridden by another SET GLOBALS statement (or GLOBALS command).

Summary of Parameters

X - Parameter may be specified dynamically / with GLOBALS command / with FORMAT statement.
S - Parameter may be specified at Statement level.
E - Parameter may be specified at Element level.
C - Parameter is evaluated at Compilation time.
R - Parameter is evaluated at Runtime.

	Default Value	Dynamic Parameter	GLOBALS Command	Statements						
				SET GLOBALS	FORMAT	DISPLAY	INPUT	REINPUT	WRITE	PRINT
AD - Attribute Definition					X	SE	SE	SE	SE	SE
AL - Alphanumeric Length for Output	none				X	SE	SE		SE	SE
BX - Box Definition	none				X	SE	SE		SE	
CC - Conditional Program Execution	OFF	X	X	R						
CD - Color Definition	NE				X	SE	SE	SE	SE	SE
CF - Character for Terminal Commands	%	X	X	R						
CO - Compiler Output	OFF	X	X							
CV - Control Variable	none					SE	SE		SE	SE
DC - Character for Decimal Point Notation	.	X	X	R						
DF - Date Format	S				X	SE	SE		SE	SE

 Copyright © Software AG 2002

	Default Value	Dynamic Parameter	GLOBALS Command	Statements						
				SET GLOBALS	FORMAT	DISPLAY	INPUT	REINPUT	WRITE	PRINT
DFOUT - Date Format for Output	S	X	X	R						
DFSTACK - Date Format for Stack	S	X	X	R						
DFTITLE - Date Format in Default Page Title	S	X	X	R						
DU - Dump Generation	OFF	X	X	R						
DY - Dynamic Attributes	none					SE	SE		SE	SE
EJ - Page Eject	ON	X	X	R						
EM - Edit Mask	none				X	SE	SE		SE	SE
ENDIAN - Endian Mode for Compiled Objects	DEFAULT	X	X							
ES - Empty Line Suppression	OFF				X	S			S	
FC - Filler Character	blank				X	SE				
FCDP - Filler Character for Dynamically Protected Fields	ON	X	X	R						
FL - Floating Point Mantissa Length	none				X	SE	SE		SE	SE
FS - Format Specification	OFF	X	X	R						
GC - Filler Character for Group Headers	blank				X	SE				
HC - Header Centering	C				X	SE				
HE - Helproutine	none						SE			

	Default Value	Dynamic Parameter	GLOBALS Command	Statements						
				SET GLOBALS	FORMAT	DISPLAY	INPUT	REINPUT	WRITE	PRINT
HW - Heading Width	ON				X	S				
IA - INPUT Assign Character	=	X	X	R						
IC - Insertion Character	none				X	SE				
ID - INPUT Delimiter Character	,	X	X	R						
IM - INPUT Mode	F	X	X	R						
IP - INPUT Prompting Text	ON				X		SE			
IS - Identical Suppress	OFF				X	SE			SE	
KD - Key Definition	OFF				X					
LC - Leading Characters	none				X	SE				
LE - Limit Error Processing	OFF	X	X	C						
LS - Line Size	physical	X	X	C	X	S	S		S	
LT - Limit of Records Read	99999999	X	X	R						
MC - Multiple-Value Field Count	1				X	S	S		S	S
ML - Msg Line Position	T (mainframe) B (Windows, UNIX, OpenVMS)	X	X							
MP - Maximum Number of Pages	32767				X	S			S	S
MS - Manual Skip	OFF				X		S			

 Copyright © Software AG 2002

	Default Value	Dynamic Parameter	GLOBALS Command	Statements						
				SET GLOBALS	FORMAT	DISPLAY	INPUT	REINPUT	WRITE	PRINT
MT - Maximum CPU Time	60 sec.	X	X	R						
NC - Use of Natural System Commands	OFF	X	X	R						
NL - Numeric Length for Output	none				X	SE	SE		SE	SE
OPF - Overwriting of Protected Fields by Helproutines	ON	X	X	R						
PC - Periodic Group Count	1				X	S	S		S	S
PD - NATPAGE Page Dataset	50	X	X	R						
PM - Print Mode	none	X	X	C	X	SE	SE		SE	SE
PS - Page Size	physical	X	X	CR	X	S	S		S	
REINP - Internal REINPUT for Invalid Data	ON	X	X	R						
SA - Sound Terminal Alarm	OFF	X	X	R						
SF - Spacing Factor	1	X	X	C	X	SE				
SG - Sign Position	ON				X	SE	SE		SE	SE
SL - Source Line Length	72	X	X							
SM - Structured Mode	OFF	X	X							
SYMGEN - Generate Symbol Tables	OFF	X	X							

	Default Value	Dynamic Parameter	GLOBALS Command	Statements						
				SET GLOBALS	FORMAT	DISPLAY	INPUT	REINPUT	WRITE	PRINT
TC - Trailing Characters	none				X	SE				
TS - Translate Output from Programs in System Libraries	OFF	X	X							
UC - Underlining Character	-				X	SE				
WH - Wait for Record in Hold Status	OFF	X	X	R						
ZD - Zero Division Check	ON	X	X	R						
ZP - Zero Printing	ON	X	X	C	X	SE	SE	SE	SE	SE

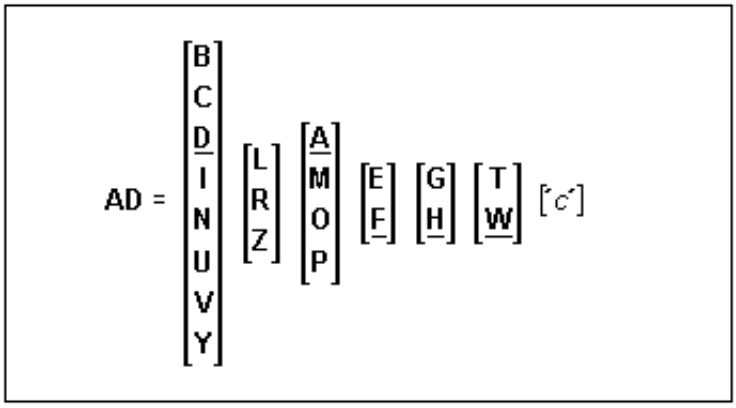
 Copyright © Software AG 2002

AD - Attribute Definition

Parameter	Possible Values	Default Value	Applicable Statements
AD	see below	see below	CALLNAT DISPLAY FORMAT INPUT PERFORM PRINT REINPUT WRITE

With this session parameter, you can specify field attributes at field or statement level.

You can specify multiple attributes in any sequence:



The meaning of each value is explained on the following pages.

Default values are underlined.

For the alignment of field values (2nd column of values in above syntax), the default value is
L - for alphanumeric fields, and
R - for numeric fields.

Examples:

```
DISPLAY #FIELDA (AD=R)
INPUT #FIELDB (AD=M)
INPUT (AD=IM) #FIELDA #FIELDB
```

Field Representation

Value	Meaning	Statements	Explanation
B	blinking (*)	all	The value of the field is displayed blinking.
C	cursive/italic (*)	all	The value of the field is displayed cursive/italic.
D	default intensity	all	The value of the field is displayed with normal intensity, that is, not highlighted in any way.
I	intensified	all	The value of the field is displayed intensified.
N	non-display	all	A value entered in the field will not be displayed.
U	underlined	all	The value of the field is displayed underlined.
V	reverse video (*)	all	The value of the field is displayed reverse video.
Y	dynamic attributes	INPUT DISPLAY WRITE	Attributes are to be controlled via a control variable (format C).

*** The field representation attributes marked with an asterisk (*) require corresponding hardware features, and will be ignored at runtime if these features are not available.**

Field Alignment

Value	Meaning	Statements	Explanation
L	left-justified	all	The value of the field is displayed left-justified.
R	right-justified	all	The value of the field is displayed right-justified.
Z	leading zeros	all	Numeric values are displayed with leading zeros, right-justified.

Field Input/Output Characteristics

Value	Meaning	Statements	Explanation
A	input field, non-protected	INPUT	The value of the field is to be entered in response to the INPUT statement.
M	output field, modifiable	INPUT CALLNAT	The value of the field is to be displayed during INPUT statement execution, and a different value may be entered by the user. The field is an output field and may be modified.
O	output field, write-protected	INPUT CALLNAT	The value of the field is to be displayed during INPUT execution. The field is an output field and may not be modified.
P	temporarily protected	INPUT REINPUT	Used in conjunction with control variable (format C), the DY parameter, and the REINPUT statement.

Mandatory Input

 Copyright © Software AG 2002

Value	Meaning	Statements	Explanation
E	value mandatory	INPUT	A value must be entered in the field in response to an INPUT statement; otherwise an error message will be issued. This is only relevant for input-only fields (AD=A).
F	value optional	INPUT	A value can, but need not, be entered in the field in response to an INPUT statement.

Length of Input Value

Value	Meaning	Statements	Explanation
G	value size	INPUT	The value entered in the field in response to an INPUT statement must be of the same length as the field. This is only relevant for input-only fields (AD=A).
H	value size	INPUT	The value entered in the field in response to an INPUT statement may be shorter than the field.

Field Upper/Lower Case Characteristics

Value	Meaning	Statements	Explanation
T	translate lower to upper case	INPUT	The value entered is to be translated to upper case.
W	accept lower case	INPUT	Lower case values are to be accepted. To make AD=W effective, you have to specify the value ON for the Natural profile parameter LC.

Filler Character

Value	Meaning	Statements	Explanation
’*c*’	filler character	INPUT	The empty field is to be filled with the specified character c (for display only) if AD=A or AD=M is specified.

Note for mainframe computers:
If the filler character is set to blank (XŸ40Ÿ), filling blanks are replaced by XŸ00Ÿ to allow for insertion of characters without having to clear the remainder of the input field before. In BS2000 environments, XŸ00Ÿ characters are displayed as dots on 97xx type terminals. Their appearance can be changed by means of the SIDA utility or with the configuration utility of the respective terminal emulation.

Copyright © Software AG 2002

AL - Alphanumeric Length for Output

Parameter	Possible Values	Default Value	Applicable Statements
AL	1 to n (n = value of LS parameter minus 1)	none	DISPLAY FORMAT INPUT PRINT WRITE

With this session parameter, you specify the default output length for an alphanumeric field; that is, when it is specified shorter than the field length, the field will be right-truncated.

It is not recommended to use the AL parameter for input fields (AD=A or AD=M) in an INPUT statement.

Any edit mask specified for a field will override the AL parameter for this field.

Example:

```
FORMAT AL=20
```

 Copyright © Software AG 2002

BX - Box Definition

Note:
This session parameter is for Mainframes only.

Parameter	Possible Values	Default Value	Applicable Statements
BX	T, B, L, R, ON, OFF	none	DISPLAY FORMAT INPUT WRITE

The outlining feature is only available on certain types of terminals, usually those which also support the display of double-byte character sets. If the terminal used does not support outlining, this parameter will be ignored at execution time.

If you use the session parameter settings BX=L or BX=R, you should switch off Natural's screen optimization using the profile parameter setting DSC=OFF or the terminal command %RO.

Outlining (boxing) is the capability to generate a line around certain fields when they are displayed on the terminal screen. Drawing such "boxes" around fields is another method of showing the user the lengths of fields and their positions on the screen.

With this parameter, you specify which parts of a box are to be displayed:

T	Top horizontal line.
B	Bottom horizontal line.
L	Lefthand vertical line.
R	Righthand vertical line.

You can specify the values in any order.

Example:

```
DISPLAY #FIELD1 (BX=RLT) /
        #FIELD2 (BX=TLRB)
```

BX=ON corresponds to **BX=TBLR.**

BX=OFF causes no boxes to be drawn around the fields concerned.

See also the terminal command %D=.

CC - Conditional Program Execution

Note:
This session parameter is for Mainframes only.

Parameter	Possible Values	Default Value	Applicable Statements
CC	ON / OFF	OFF	SET GLOBALS

With this parameter, you determine the action to be taken if an error is detected during compilation/execution of a Natural program in batch mode.

CC=ON	Natural will flush the input data stream for SYNIN and OBJIN until a line containing "%%" in the first two positions is encountered or until an end-of-file condition is detected. If more data are available in the input stream, Natural will resume reading after the line containing "%%".
CC=OFF	Natural will attempt to process the next program (or command) in the input stream.

When a Natural session terminates, return code 4 will be passed to the invoking program via register 15 if an error is detected (regardless of the CC setting).

Example:

```
SET GLOBALS CC=ON
```

 Copyright © Software AG 2002

CD - Color Definition

Parameter	Possible Values	Default Value	Applicable Statements
CD	BL = blue GR = green NE = neutral PI = pink RE = red TU = turquoise YE = yellow	NE	DISPLAY FORMAT INPUT REINPUT PRINT WRITE

With this session parameter, you specify the color in which fields are to be displayed.

Note:
If no color screen is used, this parameter will be ignored at runtime.

Example:

```
INPUT (CD=RE) #A #B
```

CF - Character for Terminal Commands

Parameter	Possible Values	Default Value	Applicable Statements
CF	any special character, OFF	%	SET GLOBALS

With this session parameter, you specify the control character for Natural terminal commands, that is, the character which is to be used as the first character of any terminal command.

For example, if the CF setting is "%" (which is the default), a terminal command must begin with the character "%".

If CF=OFF is specified, no control character for terminal commands will be available. Terminal commands issued with a SET CONTROL statement, however, will nonetheless be accepted. CF=OFF may only be specified dynamically when invoking Natural, or with a GLOBALS command, but not with a SET GLOBALS statement.

The character specified with the CF parameter must not be the same as the one specified with the HI parameter (help character) or IA parameter (input assign character).

Note:
The character specified with the CF parameter should not be the same as the one specified with the DC parameter (decimal character) or ID parameter (input delimiter character).

Example:

```
SET GLOBALS CF=+
```

Note:
In the map editor, the control character for terminal commands is always "%" (so as to avoid conflicts with delimiter characters used in maps), no matter which character is defined with the CF parameter.

 Copyright © Software AG 2002

CO - Compiler Output

Note:
This session parameter is for UNIX/OpenVMS platforms only.

Parameter	Possible Values	Default Value	Applicable Statements
CO	ON / OFF	OFF	SET GLOBALS

This parameter should not be used without prior consultation of Software AG support.

With this parameter, you determine whether a compiler listing is to be displayed when you compile a Natural object.

Setting CO=ON causes a significant longer compilation time than without using this parameter (CO=OFF). Especially a CATALL command will take a rather long time to compile all objects. Therefore this parameter should be set to OFF as default.

CO=ON	A compiler listing is displayed on the screen.
CO=OFF	No compiler listing will be displayed.

CV - Control Variable

Parameter	Possible Values	Default Value	Applicable Statements
CV	see below	none	DISPLAY INPUT PRINT WRITE

This session parameter is used to reference a control variable.

A control variable is defined with format C and is used to assign field attributes dynamically.

The following field attributes can be modified dynamically:

- Field representation attributes
  (B, C, D, I, N, U, V; see the session parameter AD).
- Field protection (P; see the session parameter AD).
- Color (BL, GR, NE, PI, RE, TU, YE; for an explanation of color codes, see the session parameter CD.

Example:

```
  DEFINE DATA LOCAL
  1 #ATTR(C)
  1 #A   (N5)
  END-DEFINE
  ...
  MOVE (AD=I CD=RE) TO #ATTR
 INPUT #A (CV=#ATTR)
  ...
```

A control variable can also be used to check whether the contents of fields have been modified during the execution of an INPUT statement.

```
  IF #ATTR MODIFIED ...
```

Note:
If the CV parameter is specified at statement level and at field level and the control variable for the individual field is empty, the control variable for the statement will be used for the field.

 Copyright © Software AG 2002

DC - Character for Decimal Point Notation

Parameter	Possible Values	Default Value	Applicable Statements
DC	any character (except numeric characters)	. (period)	SET GLOBALS

With this session parameter, you specify the character to be used for decimal point notation.

The character assigned to DC will be in effect for all notations, where a decimal point is possible, i.e. variables, constants, edit masks.

The character specified with the DC parameter must not be the same as the one specified with the IA parameter (input assign character) or ID parameter (input delimiter character).

Note:
The character specified with the DC parameter should not be the same as the one specified with the CF parameter (control character for terminal commands) or HI parameter (help character).

Example:

```
SET GLOBALS DC=,
```

DF - Date Format

Parameter	Possible Values	Default Value	Applicable Statements
DF	S, I, L	S	COMPRESS DISPLAY FETCH FORMAT INPUT MOVE PRINT RUN STACK WRITE

When the value of a date field is converted to alphanumeric format (for example, in a MOVE, DISPLAY, WRITE or INPUT statement) and no edit mask is specified for the conversion, the default date format as determined by the **profile parameter** DTFORM is used as edit mask. The same is true for the input validation of a date variable used in an INPUT statement: If no edit mask is specifed, the input is validated according to the date format determined by the DTFORM parameter.

With the DF session parameter, you determine the length of a date when converted to alphanumeric representation without an edit mask being specified:

DF=S	8-byte representation with 2-digit year component and delimiters (yy-mm-dd).
DF=I	8-byte representation with 4-digit year component and no delimiters (yyyymmdd).
DF=L	10-byte representation with 4-digit year component and delimiters (yyyy-mm-dd).

Note:
The sequence of the day, month and year components and the delimiter characters used are determined by the **profile parameter** DTFORM.

With DF=S, only 2 digits are provided for the year information; this means that if the date value contained the century, this information would be lost during the conversion.

By using DF=I or DF=L, you can gradually change your applications to use 4-digit year representations and at the same time continue to make use of the flexibility provided by the **profile parameter** DTFORM.

The DF parameter is evaluated at compilation. It can be specified with the FORMAT statement, the statements INPUT, DISPLAY, WRITE and PRINT (at statement and field level), and the statements MOVE, COMPRESS, STACK, RUN and FETCH (at field level).

See also the section Processing of Date Information in the Natural Programming Guide.

 Copyright © Software AG 2002

DFOUT - Date Format for Output

Parameter	Possible Values	Default Value	Applicable Statements
DFOUT	S, I	S	SET GLOBALS

This session parameter determines the format in which the values of date variables are displayed by INPUT, DISPLAY, PRINT and WRITE statements.

DFOUT=S	Date variables are displayed with a 2-digit year component, and delimiters as determined by the **profile parameter** DTFORM. For example: yy-mm-dd.
DFOUT=I	Date variables are displayed with a full 4-digit year component and no delimiters. For example: yyyymmdd.

The DFOUT parameter is evaluated at runtime. It applies to date fields in INPUT, DISPLAY, PRINT and WRITE statements for which no explicit edit mask is specified and for which the session parameter DF is not set.

For either DFOUT setting, the sequence of the day, month and year components in the date values is determined by the **profile parameter** DTFORM .

See also the section Processing of Date Information in the Natural Programming Guide.

DFSTACK - Date Format for Stack

Parameter	Possible Values	Default Value	Applicable Statements
DFSTACK	S, C, I	S	SET GLOBALS

This session parameter determines the format in which the values of date variables are placed on the stack via a STACK, RUN or FETCH statement.

DFSTACK=S	Date variables are placed on the stack with a 2-digit year component, and delimiters as determined by the **profile parameter** DTFORM. For example: yy-mm-dd.
DFSTACK=C	Same as DFSTACK=S. In addition, a change in the century (that is, if the century used when the value is read from the stack is not the same as the century of the original date value before it was stacked) will be intercepted at runtime.
DFSTACK=I	Date variables are placed on the stack with a full 4-digit year component and no delimiters. For example: yyyymmdd.

The DFSTACK parameter does not apply to STACK, RUN and FETCH statements for which the session parameter DF is set.

See also the section Processing of Date Information in the Natural Programming Guide.

 Copyright © Software AG 2002

DFTITLE - Date Format in Default Page Title

Parameter	Possible Values	Default Value	Applicable Statements
DFTITLE	S, L, I	S	SET GLOBALS

This session parameter determines the output format of the date in a default page title (as output with a DISPLAY, WRITE or PRINT statement).

DFTITLE=S	The date is output with a 2-digit year component and delimiters. For example: *yy-mm-dd*.
DFTITLE=L	The date is output with a 4-digit year component and delimiters. For example: *yyyy-mm-dd*.
DFTITLE=I	The date is output with a 4-digit year component and no delimiters. For example: *yyyymmdd*.

Note:
The sequence of the day, month and year components and the delimiter characters used are determined by the **profile parameter** DTFORM.

The DFTITLE parameter is evaluated at runtime. It has no effect on a user-defined page title (as specified with a WRITE TITLE statement).

See also the section Processing of Date Information in the Natural Programming Guide.

DU - Dump Generation

- For Mainframes
- For UNIX/OpenVMS and Windows

Parameter	Possible Values	Default Value	Applicable Statements
DU	ON, OFF, SNAP, FORCE	OFF	SET GLOBALS

DU for Mainframes

With this session parameter, you determine whether a memory dump is to be generated in the case of an abnormal interruption during the execution of a Natural program.

DU=OFF	No memory dump will be produced, error 954 ABEND at execution time will be produced, and subsequent action taken by Natural is determined by the setting of the CC parameter.
DU=ON	A memory dump will be produced in the case of an abnormal termination.
DU=SNAP	This will force an immediate dump in the case of an abnormal termination during a Natural session. The Natural session will continue after the dump has been taken.
DU=FORCE	This will force an immediate dump in the case of an abnormal termination during a Natural session and will terminate the Natural session immediately. This will be useful for testing purposes in some environments.

Example:

```
SET GLOBALS DU=ON
```

Notes:
Be careful when you use this parameter, because all programs and subroutines currently active for the current user will be retained in the Natural buffer pool.
DU=ON, SNAP or FORCE may cause buffer fragmentation which may result in a significant degradation in system performance.

Note for UTM:
Under UTM, this parameter is ignored; under UTM, a dump is always produced in the case of an abnormal program termination.

 Copyright © Software AG 2002

DU for UNIX/OpenVMS and Windows

With this session parameter, you determine whether a disassembled object-code dump is to be generated.

DU=ON	When one of the **system commands** CHECK, STOW, CATALOG or RUN is executed, a disassembled object-code file will be produced. This dump file will be written into the directory "tmp" below the Natural root directory. The name of the dump file consists of the source-file name and the extension ".DIA". If the source file has not been saved, the name of the dump file will be "GEN.DIA" (under OpenVMS and UNIX) or "UNTITLED.DIA" (under Windows) respectively. If the program contains database statements, dump files with the extension ".ADA" (for Adabas) or ".SQL" (for SQL databases) will also be generated. If cross-reference data are generated, a file with the extension ".XRF" will also be generated.
DU=OFF	No dump will be produced.

Example:

```
SET GLOBALS DU=ON
```

Note:
DU=ON may create a large dump file (depending on the size of the source file), which can cause a significant degradation in system performance.

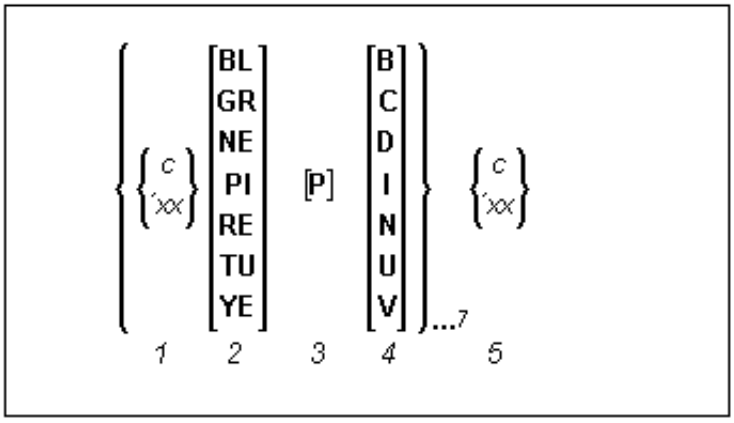
DY - Dynamic Attributes

Parameter	Possible Values	Default Value	Applicable Statements
DY	see below	none	DISPLAY INPUT PRINT WRITE

This session parameter is used to assign attributes for dynamic attribute field display.

Special identification characters (escape characters) are used to indicate the beginning and end of attribute definitions.

An alphanumeric field which is processed with an INPUT, DISPLAY, WRITE or PRINT statement, and which contains escape characters, is split into subfields at the escape character position. The corresponding attribute is then assigned to the subfield. A blank is substituted for the escape character.



1. An escape character which denotes the beginning of the attribute definition. Any special character (or a hexadecimal number preceded by an apostrophe) may be used.
2. The color attribute to be assigned
   (for explanation of color codes, see the session parameter CD).
3. Subfield is to be write-protected.
4. Additional attributes to be assigned (see the session parameter AD).
5. An escape character which denotes the end of the attribute definition. Any special character (or a hexadecimal number preceded by an apostrophe) may be used.

You may specify up to seven escape sequences (escape characters and attributes) before the character indicating the end of the attribute definitions.

Note:
For a part of a field for which a DY specification takes effect, any specification made with a CV parameter will be ignored.

Example 1:

 Copyright © Software AG 2002

```
DY = <U>
```

The text string: `THIS <is> UNDERLINED`
is printed as: `THIS` <u>`is`</u> `UNDERLINED`

Example 2:

```
DY = <BL|RE/GR>
```

Assigns: Blue to "<"
Red to "|"
Green to "/"

">" switches back to the initial field color.

Example 3:

```
DY = <P>;
```

The text string: **Do not overwrite <this>**
is printed as: **Do not overwrite this** (where "this" is protected)

EJ - Page Eject

Parameter	Possible Values	Default Value	Applicable Statements
EJ	ON / OFF	ON	SET GLOBALS

With this session parameter, you determine whether a page eject is to be performed as a result of a logical page break, a break between program input and output, and the "normal end" message.

EJ=ON	A page eject will be performed.
EJ=OFF	A page eject will not be performed. This may be used to save paper during test runs, where page ejects are not needed.

The EJ setting may be overridden using the EJECTstatement.

The EJ parameter only applies to the first report (report 0). For additional reports, the EJECT statement with report specification has to be used.

Example:

```
SET GLOBALS EJ=OFF
```

 Copyright © Software AG 2002

EM - Edit Mask

Parameter	Possible Values	Default Value	Applicable Statements
EM	see below	none	DEFINE DATA DISPLAY FORMAT INPUT MOVE EDITED PRINT WRITE

With this session parameter, you can specify an edit mask for an input and/or output field that is used in a DISPLAY, INPUT, MOVE EDITED , PRINT or WRITE statement.

For input fields, values must be entered exactly matching the edit mask. If it is desired to display the edit mask for an input field, the field should be defined as modifiable (AD=M).

For a database field, a default edit mask may have been defined in the DDM. If you specify with the EM parameter an edit mask for a database field, this edit mask specified will be used instead of any default edit mask which may be defined for the field in the DDM.

If you specify EM=OFF for a field, no edit mask will be used for the field, not even one that may be defined in the DDM.

At statement level of a DISPLAY, FORMAT, INPUT or WRITE statement, no detail field edit mask may be specified, except EM=OFF.

An edit mask overrides any settings for the session parameters AL, NL and SG.

Examples:

```
DISPLAY AA(EM=OFF) AB(EM=XX.XX)
WRITE SALARY (EM=ZZZ,ZZ9)
```

An abbreviated notation can be used for long edit masks. The following examples demonstrate the abbreviated notation which may be used for numeric, hexadecimal, and alphanumeric edit masks:

```
EM=9(4)-9(5)  is equivalent to: EM=9999-99999
EM=H(10)      is equivalent to: EM=HHHHHHHHHH
EM=X(6)..X(3) is equivalent to: EM=XXXXXX..XXX
```

This notation may be used only with the characters "9", "H", "X" and "Z" which represent significant print positions in the numeric, hexadecimal, and alphanumeric edit masks.

Blanks in Edit Masks

Blanks within an edit mask are represented by the character on your keyboard which in hexadecimal code corresponds to H’20’ (ASCII) or H’5F’ (EBCDIC) respectively, that is, the character "^" (or "¬").

Default Edit Masks

If no edit mask is specified for a field, a default edit mask is assigned to the field depending on the field format:

Field Format	Default Edit Mask
A	X
B	H
N, P, I	Z9
F	scientific representation
D	depends on default date format (as set with the **profile parameter** DTFORM)
T	HH:II:SS
L	blank / X

Edit Masks for Numeric Fields

An edit mask specified for a field of format N, P, I, or F must contain at least one "9" or "Z". If more 9s or Zs exist than the number of positions contained in the field value, the number of print positions in the edit mask will be adjusted to the number of digits defined for the field value. If fewer 9s or Zs exist, the high-order digits before the decimal point and/or low-order digits after the decimal point will be truncated.

Two options are available when using numeric edit masks. The first option allows the insertion of a special character as the first character in the edit mask.

 Copyright © Software AG 2002

Characters for the Definition of Numeric Edit Masks:

Character	Function
9	Position to be displayed (one digit of the field value).
Z	Zero suppression for leading zeros. This is the default for numeric fields. The letter "Z" may be repeatedly specified to represent floating zero suppression. A "Z" must not be specified to the right of the decimal point. A zero value may be displayed as blanks using all "Z"s in the edit mask (see also the session parameter ZP).
+	A floating sign is to be displayed preceding/following the number. The sign may be generated as a plus or minus depending on the value of the field.
-	A floating minus is to be displayed preceding/following the number if the value of the field is negative.
S	A sign is to be displayed to the left of the value. A plus sign is displayed for a positive value and a minus sign is displayed for a negative value. If "S" is used, it must be specified as the first character in an edit mask.
N	A minus sign is to be displayed to the left of the column if the value of the field is negative.
.	A period, if used as the first character, represents a decimal position and is significant.
H	Designates the beginning of a hexadecimal edit mask. If an "H" appears as the first position, all other characters which are not "H" will be considered insertion characters.

The second option allows any number of leading characters to appear prior to the first displayable position (as indicated with the digit "9"). The first of these leading characters must not be any of the edit characters described above unless enclosed in quotes. The first of these insertion characters specified will appear in the output only if the value contains leading zeros and the edit mask is defined with Z (leading zero suppression). This character will then be displayed instead of a blank for leading zeros.

Both options permit the use of insertion and trailing characters. The symbol (^) may be used to represent a leading, inserted, or trailing blank. By enclosing significant characters (9, H, Z, X) in apostrophes, it is possible to use any characters as leading, insertion, or trailing characters. Insignificant edit mask characters need not be enclosed in apostrophes. Within the same edit mask notation, it is possible to have groups of leading, insertion, and/or trailing character strings, some of which are bounded by apostrophes and some of which are not.

A trailing sign character can be specified for numeric edit masks by using the "+" or "-" characters as the last character in the edit mask. A "+" will produce a trailing "+" or "-" sign depending on the value of the field. A "-" will produce a trailing space or "-" sign depending on the value of the field. If a leading and trailing sign are specified in the edit mask, both will be produced.

Copyright © Software AG 2002

Examples of Numeric Edit Masks:

The table below lists the results obtained from the original values shown at the top of each column as they are output without editing mask. All values used as column headings represent N format fields. The lines below the top column represent the formats obtained using the different editing masks:

Value	**0000.03 (N4.2)**	**-0054 (N4)**	**+0087 (N4)**	**0962 (N4)**	**1830 (N4)**
Edit Mask					
EM=9.9	0.0	4.	7.	2.	0.
EM=99	00	54	87	62	30
EM=S99	+00	-54	+87	+62	+30
EM=+Z9	+0	-54	+87	+62	+30
EM=-9.99	0.03	-4.	7.	2.	0.
EM=N9	0	-4	7	2	0
EM=*9.99	0.03	4.	7.	2.	0.
EM=Z99	00	54	87	962	830
EM=*DMZZ9.9	DM**0.0	DM*54.	DM*87.	DM962.	DM830.
EM=999+	000+	054-	087+	962+	830+
EM=999-	000	054-	087	962	830
IC=$ EM=ZZZ.99	$.03	$54.	$87.	$962.	$830.
EM=H(6)					
- *ASCII:*	303030303033	30303574	30303837	30393632	31383330
- *EBCDIC:*	F0F0F0F0F0F3	F0F0F5D4	F0F0F8F7	F0F9F6F2	F1F8F3F0

By combining edit masks with the parameters IC and TC, negative numbers can be displayed in varying formats using a DISPLAY statement.

 Copyright © Software AG 2002

Edit Masks for Alphanumeric Fields

An alphanumeric edit mask which is only to be used with A format fields must contain at least one "X" which represents a character to be displayed. An "H" as the first character designates a hexadecimal edit mask. A blank is represented by a (^) symbol. All other characters except parentheses are permissible including leading, trailing, and insertion characters. It is also possible to specify leading, insertion, or trailing characters enclosed within apostrophes. If the character "X", a closing parenthesis, or a quotation mark is specified as an insertion character, it must be enclosed within apostrophes.

If leading characters are used before the first displayable position X, the first of these leading characters will not be displayed. Trailing characters which immediately follow the last permissible print position will be displayed.

The number of positions to be displayed will be adjusted to the length of the edit mask if the mask is shorter than the field.

Example of Alphanumeric Edit Masks:

The following program lists the alphanumeric edit masks for a field that is defined with format/length A4 and contains the value "BLUE".

```
* EXAMPLE 'EMMASK1'
DEFINE DATA LOCAL
 1 #TEXT  (A4) INIT <'BLUE'>
 END-DEFINE
WRITE NOTITLE 'MASK 1:' 5X #TEXT (EM=X.X.X.X)
       /       'MASK 2:' 5X #TEXT (EM=X^X^X^X)
       /       'MASK 3:' 5X #TEXT (EM=X--X--X)
       /       'MASK 4:' 5X #TEXT (EM=X-X-X-X-X-X)
       /       'MASK 5:' 5X #TEXT (EM=X' 'X' 'X' 'X)
       /       'MASK 6:' 5X #TEXT (EM=XX....XXX)
       /       'MASK 7:' 5X #TEXT (EM=1234XXXX)
 END
```

```
MASK 1:     B.L.U.E
MASK 2:     B L U E
MASK 3:     B--L--U
MASK 4:     B-L-U-E-
MASK 5:     B L U E
MASK 6:     BL....UE
MASK 7:     234BLUE
```

Hexadecimal Edit Masks

If the character "H" is specified as the first character in an edit mask, the content of an alphanumeric or numeric field will be displayed in hexadecimal format. Each "H" represents two print positions that will occur for each byte in the source field. Characters other than "H" serve as insertion or trailing characters in the mask. The number of positions to be displayed will be adjusted to the length of the edit mask if the mask is shorter than the field. The length of the edit mask will be adjusted to the length of the field if the field length is shorter than the edit mask.

Insertion or trailing characters may be optionally specified bounded by apostrophes.

All fields displayed with a hexadecimal edit mask are treated as alphanumeric. Therefore, if the edit mask is shorter than the field to be edited, numeric or alphanumeric positions will be displayed from left to right disregarding any decimal point positions.

Edit Mask Examples for Hexadecimal Fields:

The tables below list the hexadecimal edit masks with results obtained from the original fields and values shown above each column. All numeric values ("-10", "+10", "01") to which edit masks have been applied originated in fields defined with N2 format. The alphanumeric value "AB" originated from a field defined with format/length A2.

ASCII:

Value ->	**AB**	**-10**	**+10**	**01**
EM=HH	4142	3170	3130	3031
EM=H^H	41 42	31 70	31 30	30 31
EM=HH^H	4142	3170	3130	3031
EM=H-H	41-42	31-70	31-30	30-31
EM=H	41	31	31	30

Note:
In the case of em=h(n) (hexadecimal output), the value of n must be in the range of 1<=n<=126. If n>126, an error will be displayed only if the corresponding variable has more than 126 elements.

Example:

```
A (A100) := 'A'     /* 100 characters/elements
PRINT A (EM=H(200)) /* no error, as 100<=126, although 200>126
END
```

EBCDIC:

Value ->	**AB**	**-10**	**+10**	**01**
EM=HH	C1C2	F1D0	F1F0	F0F1
EM=H:H	C1 C2	F1 D0	F1 F0	F0 F1
EM=HH:H	C1C2	F1D0	F1F0	F0F1
EM=H-H	C1-C2	F1-D0	F1-F0	F0-F1
EM=H	C1	F1	F1	F0

 Copyright © Software AG 2002

Edit Masks for Date and Time Fields - Formats D and T

In edit masks for fields which are defined with format D (date) or T (time), the following characters can be specified:

Date - format D, and Time - format T:

Character	Usage
DD	Day.
ZD	Day, with zero suppression.
MM	Month.
ZM	Month, with zero suppression.
YYYY	Year, 4 digits.
YY	Year, 2 digits. (If used for input fields, the current century is concatenated before the input value.)
Y	Year, 1 digit. Must not be used for input fields.
WW	Number of Week.
ZW	Number of Week, with zero suppression.
JJJ	Julian day.
ZZJ	Julian day with zero suppression.
NN... or **N(n)**	Name of Day (language-dependent). The maximum length is determined by the number of N's or by n. If the name is longer than the maximum length, it will be truncated; if it is shorter, the actual length of the name will be used.
O	Number of week day (Monday = 1, Tuesday = 2, etc.). The **profile parameter** DTFORM determines whether Monday or Sunday is considered the first day of the week.
LL... or **L(n)**	Name of Month (language-dependent). The maximum length is determined by the number of L's or by n. If the name is longer than the maximum length, it will be truncated; if it is shorter, the actual length of the name will be used.
R	Year in Roman numerals (maximum 13 digits).

If only year (YY or YYYY) but no month or day is specified within an input edit mask, the values for month and day will both be set to "01". If only year (YY or YYYY) and month (MM) but no day is specified within an input edit mask, the value for day will be set to "01".

If a week is displayed which belongs to the previous or the next year, the display for year is modified accordingly.

Whether a week is week 53 of the old year or week 1 of the new year, depends on which year the Thursday of that week belongs to: if the Thursday is in the old year, the week belongs to the old year; if it is in the new year, the week belongs to the new year.

Note:
"MM" (or "ZM") and "LL" (or "L(n)") must not be specified together in an edit mask. "NN" (or "N(n)") and "O" must not be specified together in an edit mask.

Time - format T - only:

Character	Usage
T	Tenths of a second.
SS	Seconds.
ZS	Seconds, with zero suppression.
II	Minutes.
ZI	Minutes, with zero suppression.
HH	Hours.
ZH	Hours, with zero suppression.
AP	AM/PM element.

Examples of Date and Time Edit Masks:

```
 *    EXAMPLE  'EMDATI' EDIT MASKS FOR DATE AND TIME
 ********************************************************************
WRITE NOTITLE
   'DATE INTERNAL :' *DATX (DF=L) /
   '              :' *DATX (EM=N(9)' 'ZW.'WEEK 'YY)  /
   '              :' *DATX (EM=ZZJ'.DAY 'YYYY)        /
   '     ROMAN    :' *DATX (EM=R)  /
   '     AMERICAN :' *DATX (EM=MM/DD/YYYY)      12X 'OR  ' *DAT4U /
   '     JULIAN   :' *DATX (EM=YYYYJJJ)         15X 'OR  ' *DAT4J /
   '     GREGORIAN:' *DATX (EM=ZD.''L(10)''YYYY) 5X 'OR  ' *DATG ///
 *
   'TIME INTERNAL :' *TIMX                      14X 'OR  ' *TIME /
   '              :' *TIMX (EM=HH.II.SS.T) /
   '              :' *TIMX (EM=HH.II.SS' 'AP) /
   '              :' *TIMX (EM=HH)
 END
```

```
DATE INTERNAL : 2001-01-02
              : Tuesday  1.WEEK 2001
              :  2.DAY 2001
     ROMAN    : MMI
     AMERICAN : 02/01/2001           OR   01/02/2001
     JULIAN   : 200102              OR   200102
     GREGORIAN: 2.January 2001       OR   2 January 2001


TIME INTERNAL : 10:40:59             OR   10:40:59.5
              : 10.40.59.5
              : 01.40.59 PM
              : 13
```

 Copyright © Software AG 2002

Edit Masks for Logical Fields - Format L

For fields of format L (logical fields), edit masks can be defined as follows:

(**EM** = [*false-string*/] *true-string*)

The false-string must not be longer than 31 characters.

Example of Edit Masks for Logical Field:

```
 /* EXAMPLE 'EMLOGV'
 /* EXAMPLE OF LOGICAL VARIABLE IN LOGICAL CONDITION
 /*********************************************************************
DEFINE DATALOCAL
 1 #SWITCH (L) INIT <TRUE>
 1 #INDEX (I1)
 END-DEFINE
 /*********************************************************************
 FOR #INDEX 1 5
   WRITE NOTITLE #SWITCH (EM=FALSE/TRUE) 5X 'INDEX =' #INDEX
   WRITE NOTITLE #SWITCH (EM=OFF/ON) 7X 'INDEX =' #INDEX
  IF #SWITCH
   MOVE FALSE TO #SWITCH
  ELSE
   MOVE TRUE TO #SWITCH
  END-IF
 /*********************************************************************
 SKIP 1
 END-FOR
 END
```

```
TRUE      INDEX =    1
ON        INDEX =    1

FALSE     INDEX =    2
OFF       INDEX =    2

TRUE      INDEX =    3
ON        INDEX =    3

FALSE     INDEX =    4
OFF       INDEX =    4

TRUE      INDEX =    5
ON        INDEX =    5
```

ENDIAN - Endian Mode for Compiled Objects

This Natural session parameter is for UNIX/OpenVMS and Windows only.

Parameter	Possible Values	Default Value	Applicable Statements
ENDIAN	DEFAULT/LITTLE/BIG	DEFAULT	GLOBALS system command

This session parameter specifies the architecture for which the compiler should produce Natural generated programs (GP). See profile parameter ENDIAN for full details.

Example:

```
GLOBALS ENDIAN=BIG
```

 Copyright © Software AG 2002

ES - Empty Line Suppression

Parameter	Possible Values	Default Value	Applicable Statements
ES	ON / OFF	OFF	DISPLAY FORMAT WRITE

With this session parameter, you can suppress the printing of empty lines generated by a DISPLAY or WRITE statement.

If ES=ON is specified, a line resulting from a DISPLAY or WRITE statement which contains all blank values will not be printed.

ES=ON is particularly useful when displaying arrays (for example, multiple-value fields or fields contained within a periodic group) to avoid printing a large number of empty lines.

To achieve empty suppression for numeric values, the field must be specified with ZP=OFF and ES=ON in order to have null values printed as blanks.

See also the session parameters IS and ZP.

Example:

```
DISPLAY (ES=ON) NAME CITY
```

FC - Filler Character

Parameter	Possible Values	Default Value	Applicable Statements
FC	any character	blank	DISPLAY FORMAT

With this session parameter, you specify the filler character which will appear on either side of a heading produced by a DISPLAY statement across the full column width. FC only applies if the column width is determined by the field length and not by the header (see also session parameter HW); otherwise FC will be ignored.

Example:

```
DISPLAY (FC=*)
```

 Copyright © Software AG 2002

FCDP - Filler Character for Dynamically Protected Fields

Parameter	Possible Values	Default Value	Applicable Statements
FCDP	ON / OFF	ON	SET GLOBALS

With this session parameter, you can suppress the display of filler characters for input fields that have been made write-protected dynamically (that is, to which the attribute AD=P has been assigned via a control variable).

Depending on the settting of this parameter, dynamically protected input fields are displayed filled either with blanks or with the defined filler characters:

FCDP=ON	Dynamically protected input fields are displayed filled with filler characters. This may suggest to the users that they could enter something in the fields.
FCDP=OFF	Dynamically protected input fields are displayed filled with blanks.

Example:

```
DEFINE DATA LOCAL
 1 #FIELD1 (A5)
 1 #FIELD2 (A5)
 1 #CVAR1  (C) INIT <(AD=P)>
 1 #CVAR2  (C)
 END-DEFINE
 *
INPUT #FIELD1 (AD=Y'_' CV=#CVAR1)  /* field is protected
      #FIELD2 (AD=Y'_' CV=#CVAR2)  /* field is not protected
 ...
 END
```

Display with FCDP=ON: `#FIELD1 _____ #FIELD2 _____`

Display with FCDP=OFF: `#FIELD1 #FIELD2 _____`

FL - Floating Point Mantissa Length

Parameter	Possible Values	Default Value	Applicable Statements
FL	1 to 16	none	DISPLAY FORMAT INPUT PRINT WRITE

With this session parameter, you specify the mantissa length of a floating point variable during input or output.

The total length is "FL + 6" for sign, exponent, and decimal character.

```
DISPLAY FL=5     ->     +1.2345E+03
```

 Copyright © Software AG 2002

FS - Format Specification

Parameter	Possible Values	Default Value	Applicable Statements
FS	ON / OFF	OFF	SET GLOBALS

This parameter applies to reporting mode only.

This session parameter indicates whether a default format/length setting is to be in effect for the definition of user-defined variables.

FS=OFF	A user-defined variable in a Natural program for which no format/length is specified will be assigned the default format/length N7.
FS=ON	No default format/length will be assigned by Natural for a newly introduced variable in reporting mode. Format/length must be explicitly specified.

Example:

```
SET GLOBALS FS=ON
```

GC - Filler Character for Group Headers

Parameter	Possible Values	Default Value	Applicable Statements
GC	any character	blank	DISPLAY FORMAT

With this session parameter, you specify the filler character which will appear on either side of a group heading produced by a DISPLAY statement across all field columns that belong to that group. Unlike the FC parameter, which applies to individual columns, the GC parameter applies to headings across a group of columns.

Example:

```
DISPLAY (GC=*)
```

 Copyright © Software AG 2002

HC - Header Centering

Parameter	Possible Values	Default Value	Applicable Statements
HC	C, L, R	C	DISPLAY FORMAT

This session parameter determines the placement of column headers.

HC=C	Headers will be centered.
HC=L	Headers will be left-justified.
HC=R	Headers will be right-justified.

Example:

```
DISPLAY (HC=L)
```

HE - Helproutine

Parameter	Possible Values	Default Value	Applicable Statements
HE	see below	none	INPUT

With this session parameter, you specify the name of a helproutine which is to be assigned to a field.

Helproutines can be created with the Natural program editor, help maps with the Natural map editor.

The helproutine may then be invoked during processing of an INPUT statement or a map by positioning the cursor under the field and pressing the help function key (as defined with the SET KEY statement) or by entering the help character ("?" by default) into the field.

The syntax of this parameter is:

$$\mathbf{HE} = operand1 \left[, \left\{ \begin{matrix} operand2 \\ = \end{matrix} \right\} \right]_{...20}$$

Operand	Possible Structure					Possible Formats												Referencing Permitted	Dynamic Definition
Operand1	C	S				A												no	no
Operand2	C	S	A			A	N	P	I	F	B	D	T	L	C	G	O	no	no

Operand1 is the name of the helproutine or map to be invoked. The name may be a 1 to 8 character alphanumeric constant or user-defined variable. If a variable is used, it must have been previously defined. The name may contain an ampersand (&); at execution time, this character will be replaced by the current value of the Natural system variable *LANGUAGE. This feature allows the use of multi-lingual helproutines or maps.

After the helproutine name, you may specify 1 to 20 parameters (operand2) which may be passed to the helproutine. They may be specified as constants or as user-defined variables which contain the values of the parameters. If an "=" is specified as a parameter, the name of the field as defined in the map definition will be passed to the helproutine; in the case of a helproutine which is assigned to a map, "=" denotes the name of the map.

Note:
The operands must be separated either by the input delimiter character (as specified with the session parameter ID) or by a comma. A comma must not be used for this purpose, however, if the comma is defined as decimal character (with the session parameter DC).

If parameters are specified, the helproutine must begin with a DEFINE DATA PARAMETER statement which defines fields that correspond in format and length with the parameters.

If the parameter notation "=" is used to pass a field or map name, the corresponding parameter in the helproutine must be specified as A65.

 Copyright © Software AG 2002

The value of the field for which a helproutine is specified may be referenced within the helproutine. This is done by specifying a field in the DEFINE DATA PARAMETER statement which corresponds in format and length with the original field. In the block of fields defined within the DEFINE DATA PARAMETER statement, this field must always be defined behind the parameters, if present.

If the field for which a helproutine is specified is an array element, its indices may be referenced by the helproutine. To do so, you specify index parameters with format/length I2 at the end of the DEFINE DATA PARAMETER statement. You may specify up to three index parameters according to array dimensions.

Execution Of Helproutines

If a helproutine is requested - by entering a "?" in the field, or by pressing the help key (as defined with a SET KEY statement), or via a REINPUT USING HELP statement - all other data that may have been entered into fields are not assigned to the program variables until all help requests have been processed.

Note:
Only one help request per INPUT statement is possible; that is, if help is requested for more than one field (for example, by entering question marks in multiple fields), only the first help request will be executed.

Example:

```
/* MAIN PROGRAM
DEFINE DATA
1 #A(A20/1:3)
END-DEFINE
...
SET KEY PF1=HELP
...
INPUT #A (2) (HE='HELPA',=)
...
END
```

```
/* HELP-ROUTINE 'HELPA'
DEFINE DATA PARAMETER
1 #VARNAME  (A65)
1 #PARM1    (A20)
1 #VARINDEX (I2)
END-DEFINE
...
```

If a value is to be passed from a helproutine to an input field, the field must be defined as modifiable (AD=M).

HW - Heading Width

Parameter	Possible Values	Default Value	Applicable Statements
HW	ON / OFF	ON	DISPLAY FORMAT

With this session parameter you determine the width of a column output with a DISPLAY statement.

HW=ON	The width of a DISPLAY column is determined by either the length of the heading text or the length of the field, whichever is longer. This is true even if no heading text is output, either because the DISPLAY statement contains the keyword NOHDR or the DISPLAY statement is a subsequent DISPLAY (see also the DISPLAY statement).
HW=OFF	The width of a DISPLAY column is determined by the length of the field. HW=OFF only applies to DISPLAY statements which do not create headers (that is, either a first DISPLAY statement with NOHDR option or a subsequent DISPLAY statement).

Example:

```
DISPLAY (HW=OFF)
```

 Copyright © Software AG 2002

IA - INPUT Assign Character

Parameter	Possible Values	Default Value	Applicable Statements
IA	any special character	=	SET GLOBALS

With this session parameter, you determine the character to be used as the assignment character for the input parameter processing in INPUT statements in keyword/delimiter mode, or when processing data from the data stack.

The character specified with the IA parameter must not be the same as the character specified with the CF parameter (control character for terminal commands), DC parameter (decimal character) or ID parameter (input delimiter character).

Note:
The character specified with the IA parameter should not be the same as the one specified with the HI parameter (help character).

Example:

```
* Program 'SAMPLE'
DEFINE DATA LOCAL
1 #A (A1)
1 #B (A1)
END-DEFINE
INPUT #A #B
WRITE #A #B
END
```

Enter the system command **GLOBALS IM=D**
Then enter the system command **SAMPLE #A=Y,#B=X**

The program produces the following output: Y X

Enter the system command **GLOBALS IA=:**
Then enter the system command SAMPLE #B:X,#A:Y

The program produces the following output: Y X

IC - Insertion Character

Parameter	Possible Values	Default Value	Applicable Statements
IC	any character	none	DISPLAY FORMAT

With this session parameter, you specify the character string to be inserted in the column immediately preceding the value of a field output with a DISPLAY statement. The width of the output column is enlarged accordingly. You can specify a string of one to ten characters.

For numeric values, the insertion characters will be placed before the first significant digit printed.

Insertion characters may optionally be specified within apostrophes, in which case any characters can be specified. Any character string specified which contains a closing parenthesis or a quotation mark must be enclosed within apostrophes. A blank in a character string not enclosed within apostrophes is represented by the character "^".

The IC and LC parameters are mutually exclusive.

Examples:

```
DISPLAY AA(IC=*)
DISPLAY SALARY(IC='$')
```

 Copyright © Software AG 2002

ID - INPUT Delimiter Character

Parameter	Possible Values	Default Value	Applicable Statements
ID	any special character, blank	, (comma)	SET GLOBALS

With this session parameter, you specify the character to be used as a delimiter for INPUT statements in keyword/delimiter mode.

If the ID setting is to be a comma, it must be specified as ID=’,’ when using the dynamic parameter facility, because the character "," separates individual parameters.

The character specified with this parameter must not be the same as the one specified with the DC parameter (decimal character) or IA parameter (input assign character).

Note:
The character specified with the ID parameter should not be the same as the one specified with the CF parameter (control character for terminal commands) or HI parameter (help character).
The period "." should not be used as input delimiter either, because this might lead to situations in which a program termination period would be misinterpreted as input delimiter.

Examples:

```
SET GLOBALS ID=/
PARM='...,ID=',',...'
```

IM - INPUT Mode

Parameter	Possible Values	Default Value	Applicable Statements
IM	F, D	F	SET GLOBALS

With this session parameter, you determine the input mode.

IM=D	Delimiter mode.
IM=F	Forms mode.

The default mode for video terminal input is forms mode.

The IM setting may also be changed with the terminal commands %D and %F.

See the INPUT statement for additional information on delimiter mode and forms mode.

Example:

```
   SET GLOBALS IM=D
```

 Copyright © Software AG 2002

IP - INPUT Prompting Text

Parameter	Possible Values	Default Value	Applicable Statements
IP	ON / OFF	ON	FORMAT INPUT

This session parameter is used to control prompting text in INPUT statements.

IP=ON	Even if no text is specified preceding the input/output in an INPUT statement, the name of the field will be generated by default as a text element preceding the field as prompting text.
IP=OFF	No automatic prompting text will be generated for input/output fields in an INPUT statement. Only fields explicitly preceded with a text element will receive the text as prompting text.

Example:

```
FORMAT IP=OFF
```

IS - Identical Suppress

Parameter	Possible Values	Default Value	Applicable Statements
IS	ON / OFF	OFF	DISPLAY FORMAT WRITE

With this session parameter, you can suppress the printing of identical information in successive lines created by a WRITE or DISPLAY statement.

If IS=ON is specified, a value which is identical to the previous value for the field will not be displayed.

If a DISPLAY or WRITE statement is used to create multiple output lines using the VERT or "/" notation, IS=ON applies only to the first line.

The IS parameter setting can be suspended for one record by issuing the SUSPEND IDENTICAL SUPPRESS statement.

The IS parameter may be used in combination with the parameters ES and ZP to cause empty line suppression.

Example:

```
FORMAT IS=ON
```

 Copyright © Software AG 2002

KD - Key Definition

Parameter	Possible Values	Default Value	Applicable Statements
KD	ON / OFF	OFF	FORMAT

This session parameter is used to display the names assigned to the PF keys (see the SET KEY statement).

This information will always be displayed automatically in the two bottom lines of the physical screen with any output created by the INPUT, WRITE, DISPLAY, and PRINT statement.

As this display requires two lines, the logical page size (see the session parameter PS) must be reduced by two.

Note for graphical user interfaces:
If PF keys are defined, they are always displayed, regardless of the setting of this parameter. If no PF keys are defined, this parameter can be used to switch on/off the display of the ENTER key.

Example:

```
FORMAT KD=ON
```

LC - Leading Characters

Parameter	Possible Values	Default Value	Applicable Statements
LC	any character	none	DISPLAY FORMAT

With this session parameter, you can specify leading characters are displayed immediately before a field output by a DISPLAY statement. The width of the output column is enlarged accordingly.

Up to 10 characters may be specified. Leading characters may optionally be specified enclosed within apostrophes, in which case, any characters can be specified. Any character string specified which contains a closing parenthesis or a quotation mark must be enclosed within apostrophes. The character "^" is used to represent a blank in a character string not enclosed within apostrophes.

The session parameters LC and IC are mutually exclusive.

Example:

```
DISPLAY LC=*
```

 Copyright © Software AG 2002

LE - Limit Error Processing

Parameter	Possible Values	Default Value	Applicable Statements
LE	ON / OFF	OFF	SET GLOBALS

With this session parameter, you determine the action to be taken if the limit specified for processing loop execution is reached.

The limit may be either a global limit (see session parameter LT) or a limit specified for a given processing loop.

LE=OFF	The Natural program will be terminated normally at the point where the limit was reached.
LE=ON	The Natural program will be terminated and an error message will be issued when the limit is reached. LE=ON applies only to programs which are loaded from a library located in the system file FUSER, that is, library SYSTEM, or with a (library) name that does not start with the prefix "SYS".

Example:

```
SET GLOBALS LE=OFF
```

LS - Line Size

Parameter	Possible Values	Default Value	Applicable Statements
LS	2 to 250 (if a default page title is used, the minimum value is 8)	physical line size	DISPLAY FORMAT INPUT SET GLOBALS WRITE

With this session parameter, you determine the maximum number of characters permitted per line for DISPLAY, INPUT and WRITE statements.

Example:

```
DISPLAY (LS=100) 'test'
```

 Copyright © Software AG 2002

LT - Limit of Records Read

Parameter	Possible Values	Default Value	Applicable Statements
LT	0 to *n* (*n* = value of *profile parameter* LT at session start)	99999999	SET GLOBALS

With this session parameter, you determine the maximum number of records which may be read in any given processing loop in a Natural program.

This limit applies to any processing loop initiated with a READ, FIND or HISTOGRAM statement. All records read, including rejected records from a WHERE clause, are counted against this limit.

If a limit for a single processing loop (as set with a LIMIT statement or a limit notation) is higher than the limit set with the LT parameter, the limit set with the LT parameter applies.

See also the session parameter LE.

Example:

```
SET GLOBALS LT=100
```

MC - Multiple-Value Field Count

Note:
This parameter may be used in reporting mode only.

Parameter	Possible Values	Default Value	Applicable Statements
MC	1 to 191	1	DISPLAY FORMAT INPUT PRINT WRITE

With this session parameter, you determine the number of values of a multiple-value field to be output by default when the field is specified without an index in a DISPLAY or WRITE statement.

Example:

```
FORMAT MC=5
```

 Copyright © Software AG 2002

ML - Msg Line Position

Parameter	Possible Values	Default Value	Applicable Statements
ML	T (top), B (bottom)	B	GLOBALS system command

With this session parameter, you specify whether the message line is displayed on top (T) or at the bottom (B) of the screen.

Example:

```
GLOBALS ML=T
```

MP - Maximum Number of Pages

Parameter	Possible Values	Default Value	Applicable Statements
MP	1 to 99999	32767	DISPLAY FORMAT PRINT WRITE

With this session parameter, you determine the maximum number of pages to be produced for a report.

The value specified is the number of physical pages and has no effect on the starting page number used.

The program will be terminated with an error message if the MP value is exceeded.

Example:

```
FORMAT MP=1000
```

 Copyright © Software AG 2002

MS - Manual Skip

Parameter	Possible Values	Default Value	Applicable Statements
MS	ON / OFF	OFF	FORMAT INPUT

With this session parameter, you control the cursor positioning during the processing of an INPUT statement.

If MS=OFF is specified, the cursor will be positioned to the next input field as soon as the value for the current field is entered with all positions.

Example:

```
INPUT (MS=ON) #A #B
```

Note: The setting MS=ON is not supported under BS2000/OSD.

MT - Maximum CPU Time

Note :
This parameter only applies to programs executing on mainframe computers in batch mode. The limit for programs operating in interactive mode is controlled by the TP monitor in use.

Parameter	Possible Values	Default Value	Applicable Statements
MT	see below	60 (seconds)	SET GLOBALS

With this session parameter, you determine the maximum amount of CPU time which may be used by a Natural program. The value is specified in seconds.

MT=0 indicates that no CPU time limit is to be in effect.

The maximum value that can be used is determined by the operating system environment. Any value greater than the maximum will be reduced to the maximum supported by the operating system.

In system environments which do not support CPU time measurement, the limit is interpreted as elapsed time. In systems without timer support, the CPU time limit is ignored.

Example:

```
SET GLOBALS MT=5
```

 Copyright © Software AG 2002

NC - Use of Natural System Commands

Parameter	Possible Values	Default Value	Applicable Statements
NC	ON / OFF	OFF	SET GLOBALS

With this session parameter, you determine whether Natural system commands can be used during the Natural session or not.

NC=OFF	All system commands can be used.
NC=ON	System commands cannot be used - except FIN, LAST, LOGOFF, LOGON, MAINMENU, RENUMBER, RETURN, SETUP, and TECH.

Note:
Natural terminal commands and user-created commands are not affected by this parameter.

Example:

```
GLOBALS NC=OFF
```

NL - Numeric Length for Output

Parameter	Possible Values	Default Value	Applicable Statements
NL	see below	none	DISPLAY FORMAT INPUT PRINT WRITE

This session parameter determines the default input/output length for a numeric field used in a DISPLAY, INPUT, PRINT or WRITE statement.

The length is specified as nn.m, where nn represents the number of positions before the decimal point and m represents the number of positions after the decimal point. The m notation is optional. The value of m must not exceed 7. The total of n+m must not exceed 29.

If NL is set less than the field length, values are truncated. No error is produced when relevant digits are truncated.

If NL is set greater than the field length, values are expanded with blanks. No error is produced when an input field is truncated.

The NL parameter must not be specified for groups.

Any edit mask specified for a field will override the NL parameter for this field.

Example:

```
DISPLAY #AA(NL=20) #AB(NL=3.2)
```

 Copyright © Software AG 2002

OPF - Overwriting of Protected Fields by Helproutines

Parameter	Possible Values	Default Value	Applicable Statements
OPF	ON / OFF	ON	SET GLOBALS

This session parameter determines whether the content of a write-protected field (attribute AD=P) can be overwritten by a helproutine assigned to the field.

OPF=ON	A helproutine assigned to a field can overwrite the field's content, even if the field is write-protected.
OPF=OFF	Helproutines cannotoverwrite the contents of write-protected fields.

The OPF parameter only applies to the field for which a helproutine is invoked, it does not affect parameters explicitly passed to the helproutine. This means that the OPF parameter takes no effect if the field for which help is invoked is also explicitly specified as a parameter to be passed to the helproutine.

Copyright © Software AG 2002

PC - Periodic Group Count

Parameter	Possible Values	Default Value	Applicable Statements
PC	1 to 99	1	DISPLAY FORMAT INPUT PRINT WRITE

This parameter may be used in reporting mode only.

This session parameter determines the number of periodic group occurrences to be output by default if a periodic group (or a field contained within a periodic group) is specified without an index in a DISPLAY or WRITE statement.

Example:

```
FORMAT PC=5
```

 Copyright © Software AG 2002

PD - NATPAGE Page Dataset

Note:
This parameter is for Mainframes and UNIX/OpenVMS only.

Parameter	Possible Values	Default Value	Applicable Statements
PD	0 to 255	50	SET GLOBALS

This session parameter is used to specify the maximum number of pages (screens) which can be recorded with the NATPAGE utility.

If this number is exceeded, each additional screen will overwrite an already recorded screen, thus gradually overwriting the oldest screens, i.e. those recorded first (wrap-around technique).

The recorded screens are stored in the Natural system file FUSER.

For further information on the NATPAGE utility, see the terminal commands %E, %I, %O, %P and %S.

Example:

```
SET GLOBALS PD=60
```

PM - Print Mode

Parameter	Possible Values	Default Value	Applicable Statements
PM	C, D, I, N	none	DEFINE DATA DISPLAY FORMAT INPUT PRINT SET GLOBALS* WRITE

* With the SET GLOBALS statement, the Natural **profile parameter** PM can be set, not the session parameter PM.

This session parameter is used to indicate how fields are to be displayed.

PM=C	(can only be set on mainframe computers) An alternative character set is used (see the module NATPM in the Natural source library).
PM=D	(can only be set on mainframe computers) Defines DBCS fields without shift codes (see Support of Double-Byte Character Sets in the Natural Operations for Mainframes documentation).
PM=I	Field values are displayed in inverse direction; that is, from right to left (for example, for use in Middle East countries).
PM=N	No hardcopy of the display can be made.

Example:

```
LIMIT 1
  READ EMPLOYEES
  DISPLAY NOTITLE NAME
  DISPLAY NOTITLE NAME (PM=I)
  DISPLAY NOTITLE NAME
  END
```

```
          NAME
 --------------------
 MORENO
                ONEROM
 MORENO
```

Note:
More than one value may be specified.

 Copyright © Software AG 2002

PS - Page Size

Parameter	Possible Values	Default Value	Applicable Statements
PS	10 to 250, 0 (see below)	physical page size	DISPLAY FORMAT INPUT SET GLOBALS WRITE

This session parameter determines the maximum number of lines per page to be used for Natural reports created with the DISPLAY / WRITE statement.

If PS=0 is specified for the first report to be output (report 0), the physical device page size minus 1 will be used. If PS=0 is specified for reports 1 - 31, this will cause automatic newpage processing to be inhibited, that is, no automatic page break processing will be performed.

PS=0 can only be specified with a FORMAT statement.

Example:

```
FORMAT PS=40
```

REINP - Internal REINPUT for Invalid Data

Parameter	Possible Values	Default Value	Applicable Statements
REINP	ON / OFF	ON	SET GLOBALS

By default, Natural automatically issues an internal REINPUT statement if invalid data have been entered in response to an INPUT statement. With this session parameter, you can switch this mechanism off. This will allow you to handle such input errors yourself in your application.

REINP=ON	An internal REINPUT statement is issued when invalid data have been entered.
REINP=OFF	An internal REINPUT statement is not issued when invalid data have been entered.

 Copyright © Software AG 2002

SA - Sound Terminal Alarm

Parameter	Possible Values	Default Value	Applicable Statements
SA	ON / OFF	OFF	SET GLOBALS

This session parameter determines whether the terminal alarm feature is to be used.

The use of this feature requires that the terminal alarm hardware feature has been installed for the terminal, otherwise the parameter will be ignored.

SA=ON	The terminal alarm will be sounded each time the user is prompted for input by Natural.
SA=OFF	The terminal alarm feature will not be used when the user is prompted for input. However, the alarm may still be activated with the SOUND ALARM option of the REINPUT statement.

Example:

```
SET GLOBALS SA=ON
```

SF - Spacing Factor

Parameter	Possible Values	Default Value	Applicable Statements
SF	1 to 30	1	DISPLAY FORMAT SET GLOBALS

This session parameter determines the default number of spaces to be inserted between columns field values on Natural reports created with the DISPLAY statement.

The SF parameter cannot be set to "0"; that is, at least one blank character must be placed between report columns.

Example:

```
GLOBALS SF=5
```

 Copyright © Software AG 2002

SG - Sign Position

Parameter	Possible Values	Default Value	Applicable Statements
SG	ON / OFF	ON	DISPLAY FORMAT INPUT PRINT WRITE

This session parameter determines whether or not a sign position is to be allocated for a numeric field.

SG=OFF causes numeric fields with negative values to be output without a "-" sign.

Example:

```
FORMAT SG=OFF
```

If the EM parameter is specified, it overrides the SG parameter.

SL - Source Line Length

Note:
This parameter is for Mainframes only.

Parameter	Possible Values	Default Value	Applicable Statements
SL	20 to 80	72	GLOBALS system command

This session parameter determines the number of characters to be interpreted on each Natural source-code line; this only applies to EDT mode (which is activated with the **system command** EDT).

Example:

```
GLOBALS SL=74
```

 Copyright © Software AG 2002

SM - Structured Mode

Parameter	Possible Values	Default Value	Applicable Statements
SM	ON / OFF	OFF	GLOBALS system command

This session parameter determines whether structured mode has to be used.

If structured mode (SM=ON) was specified by the Natural administrator during Natural installation, this parameter cannot be set to "OFF". Otherwise with SM=OFF the mode may be changed as desired.

Example:

```
GLOBALS SM=ON
```

Note:
If Natural Security is installed, this parameter may be disabled by Natural Security to the effect that structured mode is invariably in effect for a given library.

SYMGEN - Generate Symbol Table

This session parameter is for UNIX/OpenVMS and Windows platforms only.

Parameter	Possible Values	Default Value	Applicable Statements
SYMGEN	ON, OFF	OFF	GLOBALS system command

This session parameter determines whether a symbol table is generated. See also profile parameter SYMGEN.

The symbol table contains all symbols used within a Natural program (for example, variable names). It is part of the generated program and is required for the Natural Debugger and the dialog editor.

Example:

```
GLOBALS SYMGEN=ON
```

 Copyright © Software AG 2002

TC - Trailing Characters

Parameter	Possible Values	Default Value	Applicable Statements
TC	any character	none	DISPLAY FORMAT

With this session parameter, you can specify trailing characters that are to be displayed immediately to the right of a field output with a DISPLAY statement. The width of the output column is enlarged accordingly.

Up to 10 characters may be specified. Trailing characters may optionally be specified enclosed within apostrophes, in which case any characters can be specified. Any character string specified which contains a closing parenthesis or a quotation mark must be enclosed within apostrophes.

Examples:

```
FORMAT TC=*
DISPLAY (TC='*B*')
```

TS - Translate Output from Programs in System Libraries

Note:
This parameter is for Mainframes only.

Parameter	Possible Values	Default Value	Applicable Statements
TS	ON / OFF	OFF	GLOBALS system command

This session parameter is used to translate output from Natural system libraries (that is, libraries whose names begin with "SYS") using a translate table. This may be necessary for locations which have non-standard lower-case usage (for example, Middle East countries); see also session parameter PM.

Note:
If TS=ON, the profile parameter LC=OFF and the session parameter AD=T, both of which translate input to upper case, are ignored, as they would cause undesired character translation for special character sets.

 Copyright © Software AG 2002

UC - Underlining Character

Parameter	Possible Values	Default Value	Applicable Statements
UC	any character, OFF	- (hyphen)	DISPLAY FORMAT

This session parameter determines the character that is used as underlining character for the following:

- column headings generated by DISPLAY statements;
- page titles/trailers produced by WRITE TITLE / WRITE TRAILER statements with UNDERLINED option.

If you do not wish column headers to be underlined, you have two options:

- **"UC= "** - A blank line will be output instead of underlining.
- **UC=OFF** - The field values will be output immediately below the heading line, without any blank line in between.
  You can specify UC=OFF only at the statement level of a DISPLAY statement; in this case, you cannot make any other UC specifications for individual fields in that statement.

Examples:

```
FORMAT UC=*
DISPLAY (UC= ) NAME AGE (UC=+)
```

WH - Wait for Record in Hold Status

Note:
This parameter only applies to Adabas databases.

Parameter	Possible Values	Default Value	Applicable Statements
WH	ON / OFF	OFF	SET GLOBALS

This session parameter determines the action to be taken if a required record has been placed in hold status by another user.

WH=OFF	An error message will be returned if any of these records cannot be placed in hold status.
WH=ON	The user will be placed in wait status until either the requested record becomes available, or an error message is issued due to the database system exceeding a time limit or other limit while attempting to place the record in hold status.

Example:

```
SET GLOBALS WH=ON
```

 Copyright © Software AG 2002

ZD - Zero Division Check

Parameter	Possible Values	Default Value	Applicable Statements
ZD	ON / OFF	ON	SET GLOBALS

This session parameter determines the action to be taken if an attempt is made to perform a division operation in which the divisor is "0".

ZD=ON	An error message will result if an attempt to divide by "0" is detected by Natural.
ZD=OFF	Natural will return a result of "0" for any division operation in which the divisor was "0".

Example:

```
SET GLOBALS ZD=OFF
```

ZP - Zero Printing

Parameter	Possible Values	Default Value	Applicable Statements
ZP	ON / OFF	ON	DISPLAY FORMAT INPUT PRINT REINPUT SET GLOBALS WRITE

With this session parameter you can suppress the display of a numeric field (format N, I, P or F) or time field (format T) which contains a value of all zeros.

ZP=ON	A field value which consists of all zeros will be displayed as one zero (for numeric field) or all zeros (for a time field) respectively.
ZP=OFF	A field value which consists of all zeros will **not** be displayed.

Example:

```
FORMAT ZP=OFF
```

 Copyright © Software AG 2002

Parameter Modules - Overview

For mainframes only:

This document describes the optional parameter macros which can be used in a Natural parameter module.

The selection box below provides a list of hyperlinks that enables you to jump direct to a specific parameter macro.

A Natural parameter module is created by assembling the NTPRM Macro, and optionally one or more of the following macros:

Macro	Function
NTALIAS	External names of non-Natural programs.
NTBPI	Buffer pool initialization.
NTCCTAB	Printer escape-character control table.
NTCMPO	Compilation options.
NTDB	Database types and options.
NTDCOM	NaturalX support.
NTDYNP	Control use of dynamic parameters.
NTFILE	Specification of logical files.
NTKAPRI	Support of Kanji printing.
NTOPRB	Database open processing.
NTOPT	Control of Natural Optimizer Compiler.
NTPRINT	Print file assignments.
NTRPC	Handling of remote procedure calls.
NTSCTAB	Overwriting of scanner character-type table.
NTSORT	Control of sort program.
NTSYS	Specification of set of dynamic parameters.
NTTAB	Overwriting of standard output translation table.
NTTABA1	Overwriting of EBCDIC-ASCII translation tables.
NTTABA2	Overwriting of ASCII-EBCDIC translation tables.
NTTABL	Overwriting of SYS library output translation table.
NTTAB1	Overwriting of alternate output translation tables.
NTTAB2	Overwriting of alternate input translation tables.
NTTF	Translation of database ID/file number.
NTTRACE	Define components for tracing.
NTUSER	Use of alternative parameter module.
NTUTAB1	Overwriting of lower-case/upper-case translation tables.
NTUTAB2	Overwriting of upper-case/lower-case translation tables.
NTWORK	Work files assignments.

 Copyright © Software AG 2002

NTALIAS Macro - External Names of Non-Natural Programs

If the external names of static non-Natural programs (as specified with the profile parameter CSTATIC) are different from the internal program names (as used in the CALL statements), you specify an NTALIAS macro to determine which external program name is to represent which internal program name.

For each alias program definition, you need one NTALIAS definition.

Prerequisite:
The NTALIAS macro only applies if the profile parameter RCA is set to "ON" or to "*name-list*" (in the NTPRM macro or dynamically), which controls dynamic loading of static non-Natural programs to be linked to Natural.

This document covers the following topics:

- NTALIAS Syntax
- Example of NTALIAS Macro

NTALIAS Syntax

The NTALIAS macro is specified as follows:

NTALIAS *internal-program-name,external-program-name*

Example of NTALIAS Macro

```
NTALIAS PROGRAM1,ALIAS1
NTALIAS PROGRAM2,ALIAS2
```

NTBPI Macro - Buffer Pool Initialization

The NTBI macro is used to define a Natural buffer pool. For general information on the Natural buffer pool, see Natural Buffer Pool.

This document covers the following topics:

- NTBPI Syntax
- NTBPI Parameters
- Examples of NTBPI Macros
- NTBPI Alternative

NTBPI Syntax

The NTBPI macro is specified as follows:

NTBPI TYPE=*type*,SEQ=*n*,NAME=*name*,SIZE=*nnn*,LIST=*name*,TXTSIZE=*n*

NTBPI Parameters

TYPE | SEQ | NAME | SIZE | LIST | TXTSIZE

TYPE - Type of Buffer Pool

Determines the type of the buffer pool. Possible types are:

NAT	Natural buffer pool (this is the default).
DLI	DL/I buffer pool.
EDIT	Software AG Editor buffer pool. For the Software AG Editor buffer pool, see also the profile parameter EDPSIZE.
SORT	Sort buffer pool.
MON	Buffer pool for Monitor function of SYSTP utility.
DCOM	Buffer pool to hold DCOM objects (class models, class factories, instances) separately from the "normal" Natural buffer pool.

SEQ - Sequence Number of Buffer Pool

Determines the sequence number *n* of the buffer pool.

For every buffer pool, you can define one or more backup buffer pools; that is, alternative buffer pools (of the same *type*, but with a different sequence number) which will be used if the buffer pool is not available at session initialization or cannot be allocated.

Buffer pools of the same *type* are sorted in order of sequence numbers (should two pools of the same type have the same sequence number, they will be sorted in the order in which they are specified). If a requested buffer pool is not available, the buffer pool of the same type with the next higher sequence number will be used instead. If that one is not available either, the one with the next higher number will be used, etc.

 Copyright © Software AG 2002

Possible values	0 to 9
Default sequence number	0 (that is, the buffer pool is a primary buffer pool, not a backup buffer pool).

NAME - Name of Global Buffer Pool

Applies only to global buffer pools.

Determines the name of the global buffer pool (1 to 8 characters). For a local buffer pool, the name is blank.

The NAME specification can be overridden dynamically with the profile parameter BPNAME.

SIZE - Size of Buffer Pool

Applies only to local buffer pools. Determines the size *nnn* of the buffer pool (in KB). Possible values are: 100 to 2097151. The default size is 256.

Under Com-plete: The SIZE specification of the NTBPI macro is ignored for a local buffer pool. The size is established in the buffer-pool server definition (see Installing the Natural/Com-plete Interface in the Natural Installation Guide for Mainframes.

The SIZE specification can be overridden dynamically with the profile parameter BPSIZE.

LIST - Name of Preload List to be Used

Applies only to local buffer pools of TYPE=NAT.

Determines the *name* of the preload list to be used for this buffer pool. The default is blank (that is, no preload list is to be used).

For general information on preload lists, see Preload List. Preload lists are maintained with the SYSBPM utility as described in the section Debugging and Monitoring.

The LIST specification can be overridden dynamically with the profile parameter BPLIST.

TXTSIZE - Size of Buffer Pool Text Segments

Applies to local buffer pools of the following types:

- TYPE=NAT
- TYPE=SORT
- TYPE=DLI
- TYPE=DCOM.

Determines the size *n* (in KB) of the buffer pool text segments.

Possible values	1, 2, 4, 8, 12, 16 (KB)
Default value	4.

In multi-user environments (for example, under CICS), the TXTSIZE specification only affects the very first Natural session which initializes the local buffer pool.

Under BS2000/OSD: The parameters SEQ, NAME and SIZE are ignored.

The TXTSIZE specification can be overridden dynamically with the profile parameter BPTEXT.

Examples of NTBPI Macros

```
NTBPI TYPE=NAT,SEQ=0,NAME=NATBP1
NTBPI TYPE=NAT,SEQ=1,NAME=NATBP2
NTBPI TYPE=NAT,SEQ=2,SIZE=1000
```

These examples define multiple Natural buffer pools. If the global buffer pool NATBP1 is not available, the global buffer pool NATBP2 will be used instead. If the latter is not available either, a local buffer pool with a size of 1000 KB will be used.

NTBPI Alternative

Instead of the NTBPI macro, you can also use the dynamic profile parameter BPI.

If you want to dynamically replace the primary buffer pool by a local buffer pool of 2000 KB, you can specify:

```
BPI=(NAME=' ',SIZE=2000)
or
BPNAME=' ',BPSIZE=2000
```

 Copyright © Software AG 2002

NTCCTAB Macro - Printer Escape-Character Control Table

The macro NTCCTAB is used to set up a table of printer-control sequences, which is used for printing additional reports and hardcopies.

- It is possible to either translate Natural field attributes into escape sequences or specify special characters to be translated into escape sequences.
- In addition, strings can be specified which are always sent as the first output record after an open operation or as the last output record before a close operation.
- This means that by using the right profile name, you can activate your printout either in portrait mode or in landscape. Then you can use all print features of this device by using simple attributes in Natural. This makes even bar-code printing or double-height printing possible.
- NTCCTAB defines tables which are used to recognize special characters in output fields and replace them with the defined control sequences. The macro also defines the Natural attributes which are used to insert the defined control sequences.

This document covers the following topics:

- NTCCTAB Syntax
- String Syntax for CODE, CS, CCS or CSE
- Proportional Fonts
- Example of NTCCTAB Macro

NTCCTAB Syntax

The NTCCTAB macro can be specified in three variants:

1. Variant

NTCCTAB *name*, OPN=’*xxxxx*’,CLS=’*yyyyy*’

Where

name	is the name of the profile form, that is, the DEFINE PRINTER (*n*) OUTPUT ’*nnnnn*’ PROFILE ’*name*’, which is required and which has a maximum length of 8 bytes.
OPN=’*xxxxx***’**	is optional and defines a data string (up to 250 bytes) which is sent to the printer with each open operation.
CLS=’*yyyyy***’**	is optional and defines a data string (up to 250 bytes) which is sent to the printer before each close operation.

2. Variant

NTCCTAB *name*, CODE=’*n*’, CS=’*xxxx*’

Where

CODE=’*n*’	is a character which is recognized by Natural once it appears in the output string.
CS=’*xxxx*’	is the string to replace the CODE character.

3. Variant

NTCCTAB *name*, ATR=*nnnn*, CSS=’*xxxx*’, CSE=’*yyyy*’

Where

ATR=’*nnnn*’	is the Natural internal field attribut. The name is defined by the macro’ is the string (up to 20 bytes) which is inserted before the field. Cro NAMATR.
CSS=’*xxxx*’ SS	is mandatory.
CSE=’*yyyy*’	is the string (up to 20 bytes) which is inserted after the field. CSE is mandatory.

String Syntax for CODE, CS, CCS or CSE

If the string for CODE, CS, CSS or CSE is enclosed in apostrophes, it represents a character string; if not, it represents a hexadecimal string.

Proportional Fonts

If you use proportional fonts, be sure to return to a fixed-spacing font before using tables where you need correct positioning.

Example of NTCCTAB Macro

```
NTCCTAB DBCST
NTCCTAB CODE=OE,CS=400E
NTCCTAB CODE=OF,CS=0F40
NTCCTAB ATR=P5DBCS,CSS=OE,CSE=OF
*
NTCCTAB TEST,OPN=’FIRST LINE MESSAGE’,CLS=’LAST LINE MESSAGE’
NTCCTAB CODE=’<’,CS=’ B(SOB’
NTCCTAB CODE=’>’,CS=’B(S3B ’
NTCCTAB CODE=’(’,CS=’ B(S1S’
NTCCTAB CODE=’)’,CS=’B(SOS ’
NTCCTAB ATR=P2UL,CSS=’ B&&DD’,CSE=’B&&D§’
NTCCTAB ATR=P2UL,CSS=405FF1C25084C4,CSE=5FF1C250847C
NTCCTAB ATR=P2ITAL,CSS=’ B(S1S’,CSE=’B(SOS’
NTCCTAB ATR=P1HIGH,CSS=’ B(S3B’,CSE=’B(SOB’
NTCCTAB ATR=P2RVID,CSS=’ B(S-3B’,CSE=’B(SOB’
```

 Copyright © Software AG 2002

NTCMPO Macro - Compilation Options

With the NTCMPO macro, you can specify statically several options which will be evaluated when a Natural programming object is compiled.

This document covers the following topics:

- NTCMPO Syntax
- Keyword Parameters for NTCMPO Macro
- Example of NTCMPO Macro
- NTCMPO Alternative

NTCMPO Syntax

You specify the options as keyword parameters in the NTCMPO macro as follows:

NTCMPO *keyword-parameter=value,...*

Each keyword parameter can take the value "ON" or "OFF" (GFID can also be take the value "VID"). See descriptions below.

Keyword Parameters for NTCMPO Macro

The following keyword parameters are available:

KCHECK | PCHECK | DBSHORT | PSIGNF | TSENABL | GFID | FINDMUN | V22COMP | LOWSRCE

For a general description, refer to the system command COMPOPT. For details, follow the links shown above.

Example of NTCMPO Macro

```
NTCMPO KCHECK=ON,PCHECK=ON
```

In the macro, the keyword parameters can be specified in any sequence.

NTCMPO Alternative

The following alternatives are available:

- **Using the CMPO Profile Parameter**

  At session start, you can also use the dynamic profile parameter CMPO. With the CMPO parameter, you have to specify the entire string of options enclosed in parentheses, for example:

  ```
  CMPO=(KCHECK=ON,PCHECK=ON)
  ```

- **Using the system command COMPOPT**

Within a Natural session, the NTCMPO and CMPO options can be overwritten with the system command COMPOPT.

 Copyright © Software AG 2002

NTDB Macro - Database Types and Options

The NTDB macro is only required if database management systems supported by Natural other than Adabas Version 7 are to be used.

If only Adabas Version 7 is to be used, the NTDB macro can be omitted.

An NTDB macro is used to determine which database IDs are of which type of database system.

This documents covers the following topics:

- NTDB Syntax
- NTDB Options
- Examples of NTDB Macros
- NTDB Alternative

NTDB Syntax

An NTDB macro is specified as follows:

1. A database type specification without database ID applies to all database IDs for which no other database type is explicitly specified in another NTDB macro:

   NTDB *database-type*

2. If the above form of the NTDB macro is used, it must be specified before any other NTDB macros.
3. If only one database ID is specified per database type, it is specified as follows:

   NTDB *database-type*,*database-ID*

4. If for a database type more than one database ID is specified, they must be specified in parentheses:

   NTDB *database-type*,(*database-ID*,...)

5. Optionally, you can specify one or more database options after database ID(s):

   NTDB *database-type*,*database-ID*,*option*,... NTDB *database-type*,(*database-ID*,...),*option*,...

The default *database-type* is "Adabas" (or "ADAV7" for Adabas Version 7); that is, any database ID not specified otherwise with an NTDB macro is assumed to be an Adabas Version 7 database.

Other *database-types* are, for example, "DLI", "VSAM", "DB2", etc. (see the documentation for them corresponding Natural database interface).

database-IDs must be in the range from 0 to 65535. You can specify a single database ID, a list of database ID enclosed in parentheses, or an asterisk (*) to indicate all databases.

Database ID "255" must not be specified, because it is reserved for internal use.

NTDB Options

options can be:

OPEN	This only applies only to Adabas databases for which Adabas requires an open request to be issued. If OPEN is specified for such a database, an open request is always issued (even if the ETID is blank).
READ	The database is to be read-only.
ETP	The database is to be handled by Entire Transaction Propagator.
ENTIRE	The database is to be handled by Entire DB.
ADASTAR	The database is to be handled by ADASTAR.

Each option can be reset dynamically (see below) by a corresponding "NO"-prefixed option (for example, NOOPEN, NOREAD).

Examples of NTDB Macros

Example 1 of NTDB Macros

```
NTDB VSAM,(200,201,202,203,204,205)
NTDB DLI,(197,198,199)
NTDB DB2,111
NTDB PROCESS,148
```

In this example, Databases 200 to 205 are VSAM databases, Databases 197 to 199 are DL/I databases, Database 111 is a DB2 database, Database 148 represents an Entire System Server database, and all other databases are Adabas Version 5 databases.

If multiple databases are defined for Entire System Server or for DB2, only the first one in the list is accessed by default.

Example 2 of NTDB Macros

```
NTDB ADAV6
NTDB ADAV6,17,READ
NTDB ADAV62,(200,201,202,203,204,205),ETP
NTDB DLI,39,READ
```

In this example, Databases 200 to 205 are Adabas Version 6.2 databases, Database 39 is a DL/I database, and all other databases are Adabas Version 6 databases. Databases 17 and 39 are to be read-only. Databases 200 to 205 are to be handled by Entire Transaction Propagator.

 Copyright © Software AG 2002

NTDB Alternative

To dynamically overwrite NTDB specifications, you can use the dynamic profile parameter DB. With the DB parameter, you can selectively overwrite individual NTDB specifications, without affecting other NTDB specifications.

NTDCOM - NaturalX Support

The NTDCOM macro allows you to set parameters necessary to support NaturalX.

This document covers the following topics:

- NTDCOM Syntax
- NTDCOM Parameters
- Example of NTDCOM Alias
- NTDCOM Alternative

NTDCOM Syntax

You specify the NTDCOM macro as follows:

```
NTDCOM ACTPOL=value,AUTOREG=value, SERVID=value
```

The parameters available are described below.

NTDCOM Parameters

ACTPOL | AUTOREG | SERVID

ACTPOL - Default Activation Policy for NaturalX Classes

This parameter defines the default activation policy for NaturalX (DCOM) classes. It is evaluated if a Natural class is registered as a NaturalX (DCOM) class with the REGISTER command. Possible values are:

EM	Default value. External single activation is performed.
ES	External single activation is performed.
IM	Internal multiple activation is performed.

AUTOREG - Register NaturalX Classes when Cataloging

This parameter determines whether Natural classes are to be registered as NaturalX (DCOM) classes each time they are cataloged.

ON	Registry will be updated automatically.
OFF	Default value. Registry will not be updated automatically.

Alternatively, Natural classes can be registered manually with the REGISTER command.

SERVID - Server Name Used by NaturalX

This parameter can be used to determine the server name used by NaturalX. If SERVID is not set to a value, the default server name "DEFAULT" is used.

 Copyright © Software AG 2002

Possible value is a 1 - 32-byte string. There is no default value.

Example of NTDCOM Alias

```
NTDCOM ACTPOL=EM,AUTOREG=ON,SERVID=TEST-SERVER
```

NTDCOM Alternative

Instead of the NTDCOM macro, you can also use the **profile parameter** DCOM.

NTDYNP Macro - Control Use of Dynamic Parameters

With the NTDYNP macro, you can control the dynamic specification of profile parameters.

The parameter restrictions defined by the macro NTDYNP do not apply within PROFILE or SYS parameter strings.

This document covers the following topics:

- NTDYNP Syntax
- Example of NTDYNP Macros
- NTDYNP Alternative

For general information on the use of dynamic parameters, see Dynamic Assignment of Parameter Values.

NTDYNP Syntax

Three (alternative) forms of the NTDYNP macro are possible:

1. Form

To disallow the dynamic specification of profile parameters altogether, you specify:

```
NTDYNP OFF
```

2. Form

To disallow the dynamic specification of only some profile parameters, you specify:

NTDYNP OFF**,***parameter-name*,...

All profile parameters can then be specified dynamically - except those whose *parameter-names* are specified after the keyword OFF.

3. Form

To allow the dynamic specification of only some profile parameters, you specify:

NTDYNP ON**,***parameter-name*,...

Only those profile parameters whose *parameter-names* are specified after the keyword ON can then be specified dynamically. NTDYNP can be specified multiple times, but ON or OFF must be the first value on the first NTDYNP.

 Copyright © Software AG 2002

Example of NTDYNP Macros

```
NTDYNP OFF,PARM,STACK,PSEUDO,SYS,PROFILE,

NTDYNP RELO,DU,FNAT,FUSER,FDIC,FSEC,FSPOOL
```

NTDYNP Alternative

By default, that is, if you use no NTDYNP macro, *all* profile parameters can be specified dynamically (except CSTATIC and ISIZE, which can never be specified dynamically).

Instead of the NTDYNP macro, you can also use the dynamic profile parameter DYNPARM. However, the DYNPARM parameter can only be used if the parameter module used does not contain an NTDYNP macro.

NTFILE Macro - Logical Files

NTFILE macros are used for Software AG products which have their own system files (for example, Con-nect, Natural Elite) to specify where such a system file is to be located. These products use Database ID "255" and a logical file number (FNR) in their DDMs. The NTFILE macro specifies which physical database ID (DBID) and file number (and, if applicable, password and cipher key) are associated with this logical file number.

The NTFILE macro can also be used to define a scratch-pad file with logical file number "212" see also **profile parameter** ROSY and Natural Scratch-Pad File.

If an NTFILE macro is required for a specific product, the installation documentation for this product describes which *logical-FNR* is to be specified in the corresponding NTFILE macro.

Natural records the physical file information and uses it for any Adabas calls to: DBID=255,FNR=*logical-FNR*

This document covers the following topics:

- NTFILE Syntax
- NTFILE Parameters
- Example of NTFILE Macro
- NTFILE Alternative

NTFILE Syntax

An NTFILE macro is specified as follows:

```
NTFILE  ID=logical-FNR,DBID=physical-DBID,FNR=physical-FNR,PASSW=password,
  CIPH=cipher-key,OPT=RO
```

NTFILE Parameters

The following parameters are available: ID | DBID | FNR | PASSW | CIPH | OPT

ID - Logical File Number

The logical file number must be in the range from 1 to 251.

DBID - Database Identification

The DBID must be in the range from 0 to 254, or 256 to 65535.

FNR - Physical File Number

The physical file number must be in the range from 1 to 65535.

PASSW - Password

The password must be a value of 1 to 8 characters.

CIPH - Cipher Key

 Copyright © Software AG 2002

The cipher key must be a value of 8 characters.

OPT - Option

OPT=RO indicates that the file is "read-only" and is only specified if modifications on the file are to be disabled.

Example of NTFILE Macro

```
NTFILE ID=222,DBID=21,FNR=12
```

NTFILE Alternative

The NTFILE macro corresponds to the dynamic profile parameter LFILE.

NTKAPRI Macro - Kanji Printing

The NTKAPRI macro supports the printing of Kanji output. With the this macro, you specify the list of parameters for Kanji printing.

This document covers the following topics:

- NTKAPRI Syntax
- NTKAPRI Alternative

For further information, please refer to the Kanji print interface documentation (available from Software AG of Far East).

NTKAPRI Syntax

The NTKAPRI macro is specified as follows:

NTKAPRI ’*parameter*,*parameter*,*parameter*,...’

If multiple NTKAPRI macros are specified, the parameter lists are concatenated to constitute a single list.

NTKAPRI Alternative

Instead of the NTKAPRI macro, you can also use the dynamic profile parameter KAPRI.

 Copyright © Software AG 2002

NTOPRB Macro - Database Open Processing

The NTOPRB macro can be used as an alternative to the profile parameter OPRB in the NTPRM macro. The maximum length of an OPRB parameter specification is 256 bytes. If you require a longer specification, use the NTOPRB macro instead of the OPRB parameter.

This document covers the following topics:

- NTOPRB Syntax
- Examples of NTOPRB Macros
- NTOPRB Alternative

NTOPRB Syntax

The syntax of the NTOPRB macro is as follows:

NTOPRB *dbid*,'ACC=*fnr*,UPD=*fnr*'

For possible values, see the OPRB parameter; if you use Natural with VSAM, see also the Natural for VSAM documentation.

Examples of NTOPRB Macros

```
NTOPRB 12,'ACC=40,UPD=20'

    NTOPRB 15,'EXU=1,','2,3'
```

If the OPRB string for one DBID is very long, it can be divided in two or more strings separated by commas (see the second example), as the Assembler allows single strings up to 256 bytes only.

NTOPRB Alternative

If you wish to make OPRB specifications that are to apply to all databases, it is strongly recommended that you use the OPRB parameter in the NTPRM macro (and not an NTOPRB macro).

NTOPT Macro - Control of Natural Optimizer Compiler

This macro is only used if the Natural Optimizer Compiler is installed. It is used to activate/deactivate the Natural Optimizer Compiler and controls various options related to it.

Instead of the NTOPT macro, you can also use the dynamic profile parameter OPT.

See the Dynamic Profile Parameter OPT for further information.

 Copyright © Software AG 2002

NTPRINT Macro - Print File Assignments

With NTPRINT macros, you define the print files to be used during the session. Within a session, up to 31 logical print files (numbered 1 to 31) can be used.

For an online session, all print files to be used have to be defined with an NTPRINT macro (or with the corresponding dynamic profile parameter PRINT).

The software components for accessing print files in different environments are called *access methods*. For the duration of a Natural session, each logical print file can be attached to only one access method. The access method for a print file is determined by the keyword parameter AM (see below).

In a batch session, any print files not attached to a specific access method will automatically be detected by the standard batch access method, provided that they have been predefined in the JCL.

Alternatively, OS/390 batch print files need not be predefined in the JCL. They can be defined with the subparameter AM=STD of the NTPRINT macro or PRINT parameter and allocated dynamically during the session by a Natural program using the DEFINE statement or user exit USR2021.
See also FAMSTD - Overwriting of Print and Work File Access Method Assignments.

This document covers the following topics:

- NTPRINT Syntax
- NTPRINT Keyword Parameters for All Environments
- NTPRINT Keyword Parameters for AM=STD in All Environments
- NTPRINT Keyword Parameters for AM=STD in OS/390 Environments
- NTPRINT Keyword Parameters for AM=STD in VSE/ESA Environments
- NTPRINT Keyword Parameters for AM=STD in BS2000/OSD Environments
- NTPRINT Keyword Parameters for AM=COMP
- NTPRINT Keyword Parameters for AM=IMS
- NTPRINT Keyword Parameters for DEFINE PRINTER Statement
- NTPTINT Alternative

NTPRINT Syntax

In an NTPRINT macro, you first specify one or more logical print file numbers, and then several keyword parameters, which define the characteristics which are to apply to these print files:

```
NTPRINT (print-file-numbers),keyword-parameters,...
```

print-file-numbers

The file numbers must be specified first and enclosed in parentheses. The numbers can be from 1 to 31. They can be specified in any sequence. Multiple numbers must be separated from one another by commas or blanks. To specify a range of numbers, you can use a hyphen (-).

keyword-parameters

The various types of keyword parameters are described below.

For print files with different characteristics, you specify different NTPRINT macros.

Examples:

```
NTPRINT (2,12,18),AM=STD,DEST='PRINT**',OPEN=INITOBJ,CLOSE=CMD NTPRINT
  (1,3,6-11,15),AM=NAF
```

NTPRINT Keyword Parameters for All Environments

The following keyword parameters are available: AM | DEST | OPEN | CLOSE

AM - Type of Access Method

AM=*xxx* specifies the type of access method to be used:

STD	Standard sequential batch files (batch, TSO, TIAM, CMS OS simulation).
COMP	Com-plete print files.
NAF	Natural Advanced Facilities.
PC	Entire Connection.
CMS	CMS disk and SFS files.
USER	Third-party vendor print interface.
OFF	Unassigned.

Note:
PRINT=OFF is equivalent to: PRINT=((1-31)), AM=OFF).
It does not affect any of the other keyword specifications.
The output destination is specified by parameter DEST.

DEST - External Dataset Name

DEST=*name* specifies the external dataset name (1 - 8 characters).

This corresponds to the OUTPUT value of the DEFINE PRINTER statement (and can be overwritten by a DEFINE PRINTER OUTPUT specification).

The meaning of this keyword parameter depends on the access method. For AM=STD, it is the logical dataset name (DDNAME).

If the destination is to be for multiple files, two asterisks (**) have to be specified for the file number. These will then be replaced by the corresponding logical file number for each print file.

The default value of DEST depends on the access method. For AM=STD, the default is DEST='CMPRT**' (note that a DEST value including two asterisks must be enclosed in apostrophes when using it as a dynamic parameter).

 Copyright © Software AG 2002

Under CICS	There is no DEST default value for print files under CICS. Here, the DEST parameter is mandatory, that is, CICS print files without a valid DEST specification are ignored.
Under CMS	For usage of DEST under CMS, refer to Natural under VM/CMS (in the Natural Operations documentation for mainframes).
Under VSE/ESA	Only 7-character names are supported.

OPEN - Time of File Opening

OPEN=*xxx* determines when the file is to be opened:

Option	The file is opened
INIT	for output at session initialization.
OBF	sets the default OPEN value for the different environments (Batch, CICS, Com-plete, TSO).
OBJ	The file is opened when the execution of the first object which accesses the file starts (this is the general default, besides for AM=COMP and AM=IMS).
OBJ1	The file is opened when the execution of the first object on level 1 which accesses the file starts. Otherwise, it is opened when it is first accessed.
ACC	The file is opened when it is first accessed by a statement (this is the default for AM=COMP and AM=IMS).
INITOBF	for output at session initialization. Any subsequent re-opening of the file sets the default OPEN value for the different environments (Batch, CICS, Com-plete, TSO).
INITOBJ	for output at session initialization. Any subsequent re-opening of the file will be performed when the execution of the first object which accesses the file starts.
INITOBJ1	The file is opened when the execution of the first object on level 1 which accesses the file starts. Otherwise, it is opened when it is first accessed.
INITACC	for output at session initialization. Any subsequent re-opening of the file will be performed when it is first accessed by a statement.

CLOSE - Time of File Closure

CLOSE=*xxx* determines when the file is to be closed:

Option	The file is closed
OBJ	when processing of the object in which it was first accessed is finished.
CMD	when command mode, NEXT mode or MAINMENU is reached (this is the default for AM=NAF, AM=COMP and AM=IMS).
FIN	at session end (this is the default for AM=STD). With CLOSE=FIN, a DEFINE PRINTER or CLOSE PRINTER statement does not cause the print file to be closed.
CLOSE=USER	This value specifies that a print file is closed only if the file is open and one of the following conditions is true: • a CLOSE PRINTER statement is issued, • a DEFINE PRINTER statement is issued, • at session termination.

NTPRINT Keyword Parameters for AM=STD in All Environments

The following keyword parameters are available: RECFM | BLKSIZE | LRECL | TRUNC | PAD

RECFM - Default Record Format of Dataset

RECFM=*xxxx d*etermines the default record format of the dataset.

The following formats are supported:

F	Fixed
V	Variable
U	Undefined
B	Blocked
S	Spanned
A	ASA
M	Machine control characters

The following values and also combinations of values are possible:

Possible value:	F, FA, FM, FB, FBA, FBM, V, VA, VM, VB, VBA, VBM, VBS, VBSA, VBSM, U, UA, UM.
Default value:	RECFM=VBA (variable blocked with ASA).

The RECFM specification only applies if no record format is predefined in the JCL or in the dataset DCB (OS/390 only).

BLKSIZE - Default Block Size of Dataset

BLKSIZE=*nnnnn* determines the default block size (in bytes) of the dataset.

Possible values:	0, or 8 to 32767.
Default value:	1016.

The BLKSIZE specification only applies if no block size is predefined in the JCL or in the dataset DCB (OS/390 only).

LRECL - Default Record Length of Dataset

LRECL=*nnn d*etermines the default record length (in bytes) of the dataset.

Possible values:	0, or 5 - 254.
Default value:	0

 Copyright © Software AG 2002

This parameter is used particularly to check for truncation and padding.

- For RECFM=V (B) the LRECL value includes a 4-byte record descriptor word.
- If LRECL = 0 is defined, the following applies:
- With RECFM = V (B), LRECL defaults to the minimum of BLKSIZE-4 and 254.
- With RECFM = U, LRECL defaults to BLKSIZE.
- With RECFM = F (B), the maximum record length in the Natural program being executed is taken when the file is opened. If no record length from a program is available when the file is opened, for example with OPEN=INIT, a record length of 132 is taken (plus 1 for ASA or a machine control character and/or plus 4 for a record-descriptor word if the record format is variable).

The LRECL specification only applies if no record length is predefined in the JCL or in the dataset DCB (OS/390 only).

TRUNC - Truncation of Output Records

TRUNC=ON/OFF determines whether the output records are truncated:

ON	Output records that are longer than the record length (LRECL) of the dataset will be truncated (this is the default).
OFF	Error NAT1512 will be issued if an output record is longer than the dataset record length.

PAD - Padding of output records

PAD=ON/OFF determines whether the output records are padded with blanks (applies only to datasets of fixed record length):

ON	Output records that are shorter than the record length (LRECL) of the dataset will be padded with blanks (this is the default).
OFF	Error NAT1510 will be issued if an output record is shorter than the dataset record length.

NTPRINT Keyword Parameters for AM=STD in OS/390 Environments

The following keyword parameters are available: REREAD | FREE | ASA | BUFNO | DISP | VMAX

REREAD - Option for Closing of Tape File

REREAD=*xxx* sets the REREAD option for the closing of the tape file:

ON	The REREAD option is set for the CLOSE SVC. This causes the volume to be repositioned to reprocess the dataset (this is the default).
OFF	The REREAD option is not set for the CLOSE SVC.

FREE - Dataset De-allocation at File Closure

FREE=*xxx* determines whether the dataset is de-allocated when the file is closed:

ON	The FREE option is set for the CLOSE SVC, which means that the dataset is de-allocated when it is closed (and not at step termination).
OFF	The FREE option is not set for the CLOSE SVC (this is the default).

ASA - Use of ASA Record Format

ASA=*xxx* determines whether the ASA record format is used.

ON	Enforces ASA record format, regardless of the RECFM setting in the DCB or the RECFM subparameter.
OFF	Default value.

BUFNO - Default Number of OS/390 I/O Buffers of Dataset

BUFNO=*nnn* defines the default number of OS/390 I/O buffers of the dataset.

Possible values	0 - 255.
Default value	0. In this case, OS/390 allocates five I/O buffers per default.

The number of I/O buffers can improve the performance of print file access dramatically. Note that the storage for I/O buffers is allocated below the 16 MB line.

The BUFNO specification applies only if the BUFNO value is predefined in the JCL.

DISP - Open Print File for Modification

DISP=MOD determines whether the print file is opened for modification, i.e. the new records are added at the end of the file.

This corresponds to the JCL DD statement subparameter DISP=MOD.

VMAX - Enforce LRECL=BLKSIZE-4 for Variable Record Format

VMAX=*xxx* enforces LRECL=BLKSIZE-4 for variable record format.

ON	Providing a nonzero BLKSIZE value exists for the file, VMAX=ON enforces LRECL=BLKSIZE-4 for variable record format, regardless of the LRECL setting in the DCB or the LRECL subparameter.
OFF	Default value.

NTPRINT Keyword Parameters for AM=STD in VSE/ESA Environments

The following keyword parameters are available: SYSNR | LABEL | REWIND

SYSNR - Logical VSE SYS Number

SYSNR=*nn* determines the logical VSE SYS number.

 Copyright © Software AG 2002

Possible values:	1 - 99.
Default value:	By default, the SYS number is print file number plus 40; for example, VSE default SYS number for print file 11 is 11 + 40 => SYS051.

LABEL - Tape Label Processing

LABEL=*xxx d*etermines the tape label processing:

ON	The tape is in standard label format (this is the default).
OFF	The tape is unlabeled with front tape mark.
NOTM	The tape is unlabeled without front tape mark.

REWIND - Action at File Closure

REWIND=*xxx* determines the action to be taken when a tape file is closed:

ON	The tape is rewound when the file is closed (this is the default).
OFF	The tape is not rewound when the file is closed.
UNLOAD	The tape is unloaded when the file is closed.

NTPRINT Keyword Parameters for AM=STD in BS2000/OSD Environments

The following keyword parameter is available: DISP

DISP - File Open Mode

DISP=*xxx d*etermines the open mode of the file:

EXT	The open mode is set to EXTEND.
All user values	The open mode is set to the default value OUTPUT.

NTPRINT Keyword Parameters for AM=COMP

The following keyword parameter is available: DRIVER

DRIVER - Name of Com-plete Print Driver

DRIVER=*name* specifies the *name* of the Com-plete print driver to be used.

NTPRINT Keyword Parameters for AM=IMS

The following keyword parameter is available: DRIVER

DRIVER - Name of Natural IMS Print Driver

DRIVER=*name* specifies the *name* of the Natural IMS print driver to be used.

For possible values, see NIMPARM Macro Parameters and Support of the Natural WRITE (*n*) Statement in the section Natural under IMS/TM in the Natural TP Monitor Interfaces documentation.

NTPRINT Keyword Parameters for DEFINE PRINTER Statement

With the following keyword parameters, you can set default values for the DEFINE PRINTER statement options of the same names (see the Natural Statements documentation). When a printer is closed, all DEFINE PRINTER statement options are reset to their default values.

The following keyword parameters are available: PROFILE | NAME | FORMS | DISP | COPIES | CLASS | PRTY

PROFILE - Name of Printer Control Characters Table

PROFILE=*name* specifies the *name* of printer control characters table (NTCCTAB macro).

NAME - Name of Listing

NAME=*name* specifies the listing *name*.

FORMS - Name of Listing Forms

FORMS=*name* specifies the listing forms *name*.

DISP - Listing Disposition

DISP=*disposition* specifies the listing *disposition* (HOLD, KEEP, DELETE or LEAVE).

COPIES - Number of Copies

COPIES=*nnn* specifies the number of copies to be printed (1 - 255).

CLASS - Spool Class

CLASS=*class* specifies the spool class (1 byte).

PRTY - Listing Priority

PRTY=*nnn* specifies the listing priority (1 - 255).

NTPRINT Alternative

Instead of the NTPRINT macro, you can also use the dynamic profile parameter PRINT. With the PRINT parameter, you have to specify the entire string of values enclosed in parentheses.

 Copyright © Software AG 2002

Example:

```
PRINT=((1,2,5-12,18),AM=STD,OPEN=INITOBJ,CLOSE=OBJ)
```

NTPRM Macro - Create Natural Parameter Module

The NTPRM macro must be assembled in order to create a Natural parameter module.

Generally, you can use the default values of the profile parameters in the NTPRM macro. If any of the default values do not suit your requirements, you can overwrite them with your own values.

For a description of the individual profile parameters, refer to Profile Parameters.

NTPRM Syntax

The syntax for this macro is:

```
NTPRM  parameter=value,...
```

Assembler Macro Coding Conventions

Assembler macro coding conventions must be adhered to when changing parameter values.

For example,

- the first entry must begin in column 2 or beyond and cannot extend beyond column 71;
- continuation to another line is accomplished by placing a comma after the last entry, inserting a non-blank character in column 72 and continuing the entry on the next line starting in column 16;
- a parameter and its value must always be entered on the same line.

 Copyright © Software AG 2002

NTRPC Macro - Handling of Natural Remote Procedure Calls

In this macro, you can specify keyword parameters which affect the handling of Natural RPC. For dynamic specification of the keyword parameters described below, you can use the profile parameter RPC.

This document covers the following topics:

- NTRPC Syntax
- NTRPC Macro Keyword Parameters that Apply to Client and Server
- NTRPC Macro Keyword Parameters that Apply to the Server Only
- NTRPC Macro Keyword Parameters that Apply to the Client Only

For additional information, see the Natural Remote Procedure Call documentation.

NTRPC Syntax

The NTRPC macro is specified as follows:

```
NTRPC keyword_parameter=value,...
```

keyword parameter - there are three groups of keyword parameters available that apply to

- both Client and Server
  (ACIVERS | MAXBUFF | RPCSIZE | SERVER)
- the Server only
  (LOGONRQ | NTASKS | SRVNAME | SRVNODE | SRVUSER | TRACE | TRANSP)
- the Client only
  (ACIPATT | AUTORPC | COMPR | DFS | RDS | TIMEOUT | TRYALT)

These keyword parameters are described below.

Example:

```
NTRPC RPCSIZE=80,
      SRVNAME=MYSERV,
      SERVER=ON,
      DFS=(SRV2,NODE1,,ACI),
      RDS=((SRVX,NODEX),(SRVY,NODEY)
```

NTRPC Macro Keyword Parameters that Apply to Client and Server

The following keyword parameters are available: ACIVERS | MAXBUFF | RPCSIZE | SERVER

ACIVERS - Define ACI Version for Use with EntireX Broker ACI

ACIVERS=*n* specifies the API version to be used with the EntireX Broker ACI. The broker stub in use must support the ACI version defined here. The setting of ACIVERS enables special features of the EntireX Broker depending on the API version you are using. For details see Setting Up an EntireX Broker Access in the Natural Remote Procedure Call documentation.

Possible settings	1 - 5	Single-digit number. The higher the version, the more features are available.
Default setting	2	API version 2 is used.
Dynamic specification	YES	
Specification within session	NO	

MAXBUFF - Size of Buffers

MAXBUFF=*nn* can be specified on both the client and the server side.

On the server side it determines the size of the buffer provided by the server to receive the client request including data and to send back the result.
On the client side, it determines the size of the buffer provided for the automatic execution of Natural RPC calls. This buffer is used to build the client request including data and to receive the result. For further information, see Stubs and Automatic RPC Execution in the Natural Remote Procedure Call documentation.

Possible settings	0-16380, but smaller than or equal to RPCSIZE-4	Maximum buffer size in KB. Note that the transport layer you are using may have specific restrictions. **Note:** For compatibility reasons with earlier Natural versions the value 32 is truncated from 32K to 32000 bytes. This allows you to specify MAXBUFF=32 together with ACIVERS=1.
Default setting	0	No buffer is allocated.
Dynamic specification	YES	
Specification within session	NO	

Note:
If you are using EntireX Broker, the value of MAXBUFF must not be greater than the EntireX Broker attribute MAX-MSG. Additionally, if EntireX Broker uses Entire Net-work, MAXBUFF is restricted by the startup parameter IUBL and must not exceed 32 KB. However, a maximum of 30 KB is recommended.

RPCSIZE - Size of Buffer Used by Natural RPC

RPCSIZE=*nnnnn* determines the size of the buffer used by Natural RPC.

Possible settings	0 - 16384	Buffer size *nnnnn* in KB.
Default setting	0	Natural RPC cannot be used.
Dynamic specification	YES	
Specification within session	NO	

 Copyright © Software AG 2002

Support of Parameter Name SIZE

The earlier parameter name SIZE is still supported for compatibility reasons. It will be dropped in the next version.

SERVER - Start Natural Session as an RPC Server Session

SERVER=ON/OFF specifies whether or not the Natural session will be started as an RPC server session.

Possible settings	ON	The Natural session will be started as a RPC server session.
	OFF	The Natural session will not be started as a RPC server session.
Default setting	OFF	
Dynamic specification	YES	
Specification within session	NO	

NTRPC Macro Keyword Parameters that Apply to the Server Only

The following keyword parameters are available: LOGONRQ | NTASKS | SRVNAME | SRVNODE | SRVUSER | TRACE | TRANSP

LOGONRQ - Logon for RPC Server Request Required

LOGONRQ determines whether or not a logon is required for an RPC server request. For further information, see Using Natural RPC with Natural Security in the Natural Remote Procedure Call documentation.

Possible settings	ON	A logon is required; that is, the server only accepts requests from clients which include logon data in the RPC server request. For conversational requests, the logon data is only necessary when the conversation is opened.
	OFF	A logon is not *required*. Logon data is nevertheless processed.
Default setting	ON	
Dynamic specification	YES	
Specification within session	NO	

For Natural clients the logon data can either be requested:

- by setting the LOGON option of the SYSRPC Service Directory Maintenance
- using the logon indicator of the profile parameter DFS.

NTASKS - Number of Server Tasks to be Started

NTASKS=*nn* specifies the number *nn* of server tasks to be started.

If the server has to handle a large number of client requests, you can use this option to improve the throughput by starting multiple (identically named) copies of the same server process.

Possible values:	1 - 99	Number of copies of one server process.
Default value:	1	
Dynamic specification	YES	
Specification within session	NO	

SRVNAME - Name of RPC Server

It specifies the name of the RPC server, with which it registers on the node specified with the profile parameter SRVNODE.

Possible settings	1 - 8 characters	Valid server name.
Default setting	none	
Dynamic specification	YES	
Specification within session	NO	

SRVNODE - Name of Node

SRVNODE=*node* specifies the name of the node upon which an RPC server registers.

Possible settings	1 - 8 characters	Node name.
Default setting	none	
Dynamic specification	YES	
Specification within session	NO	

SRVUSER - Server User ID for RPC Server Registry

SRVUSER=*user-id* specifies the user ID needed to register an RPC server on the node specified with the profile parameter SVRNODE. If this specification is omitted, the timestamp will be used.

Possible settings	1 - 16 characters	User ID. If SRVUSER is set to *NSC and Natural Security is installed, the Natural server uses the current Natural user ID (*USER) and the password defined for this user ID in Natural Security to logon to the node.
Default setting	timestamp	
Dynamic specification	YES	
Specification within session	NO	

TRACE - Activate RPC Trace Facility

TRACE=*n* activates the RPC trace facility and determines the trace level *n* to be used. For further information, see Using the Server Trace Facility p.p. in the Natural Remote Procedure Call documentation.

 Copyright © Software AG 2002

Possible settings	0	Nothing is traced.
	1	Only messages (inclusive Natural errors) are traced.
	2	All messages and data from/to client are traced.
Default setting	0	
Dynamic specification	YES	
Specification within session	NO	

The values 3-9 displayed in the selection box are for future use. If one of these values is chosen, then TRACE=2 is selected.

TRANSP - Server Transport Protocol

It determines which server transport protocol is used. If ACI is used, you can additionally specify the transport method.

Possible settings	ACI	ACI is used. The transport method is defined by the EntireX Broker.
	CSCI	CSCI is used. (OpenVMS only).
	ACI,TCP	ACI is used with TCP/IP.
	ACI,NET	ACI is used with Entire Net-work, i.e. using the Adabas protocol.
	ACI,TCP-NET	Trying to use ACI with TCP. If not available, ACI is used with NET.
	ACI,NET-TCP	Trying to use ACI with NET. If not available, ACI is used with TCP.
Default setting	ACI	
Dynamic specification	YES	
Specification within session	NO	

NTRPC Macro Keyword Parameters that Apply to the Client Only

The following keyword parameters are available: ACIPATT | AUTORPC | COMPR | DFS | RDS | TIMEOUT | TRYALT

ACIPATT - Define Node Pattern for ACI Protocol

ACIPATT=*node-pattern* determines which node names will be used for the ACI transport protocol together with Natural Remote Procedure Call. You can specify a pattern to match node names used for the ACI transport protocol (that is EntireX Broker nodes).

It applies only to nodes which were defined in the server maintenance of the SYSRPC utility (because there the transport protocol cannot be specified).

Possible settings	1-8 characters, including *	Pattern for node name matching. All node names matching this pattern are assumed to be ACI nodes.
Default setting	*	All node names match the pattern. The ACI transport protocol is used for all of them.
Dynamic specification	YES	
Specification within session	YES	At runtime, this value can be overwritten by the parameter maintenance of the SYSRPC utility.

Node names beginning with the letter(s) specified and/or up to the first "*" are matched. For example, if you specify N*, all node names beginning with N are regcognized as ACI nodes.

AUTORPC - Automatic Natural RPC Execution

This parameter determines whether or not Natural RPC will automatically try to execute a subprogram remotely (on the server side) which was not found locally (on the client side). For details see Stubs and Automatic RPC in the Natural Remote Procedure Call documentation.

Possible settings	ON	Natural RPC will automatically try to execute it remotely.
	OFF	Natural RPC will not automatically try to execute it remotely. If AUTORPC=OFF, you can execute CALLNATs remotely using stubs.
Default setting	OFF	
Dynamic specification	YES	
Specification within session	NO	

End of AUTO Parameter Name Support

The earlier parameter name AUTO is still supported for compatibility reasons. It will be dropped in the next version.

COMPR - Set RPC Buffer Compression

This parameter can be used to set the RPC buffer compression. It is effective only, if the automatic Natural RPC execution is used (AUTORPC=ON) and the CALLNAT is executed without a stub. If a stub is used, the compression has already been set during stub generation. For details see Using Compression in the Natural Remote Procedure Call documentation.

 Copyright © Software AG 2002

Possible settings	0	No compression will be performed.
	1	The send buffer contains modifiable fields and output fields and the format buffer. The reply buffer contains modifiable fields and input fields.
	2	Same as COMPR=1, additionally the reply buffer also contains the format buffer.
Default setting	1	
Dynamic specification	YES	
Specification within session	YES	At runtime, this value can be overwritten by the parameter maintenance of the SYSRPC utility.

DFS - RPC Default Server Address

This parameter can be used to define an RPC default server address. It determines the server name, the server node, the logon indicator and the transport protocol. The default server address will be used only if no appropriate server is found in the service directory. For further information, see Specifying RPC Server Adresses in the Natural Remote Procedure Call documentation.

To define a default server address you specify up to 4 subparameters.

Possible settings	4 subparameters	The *server name* (1 - 8 characters). There is no default, the value must be specified.
		The *server node* (1 - 8 characters). There is no default, the value must be specified.
		A *logon indicator*. If nothing is specified, blank is the default. L The client initiates a Natural logon to the server with the library name of the current library on the client. blank No server logon will be executed. Note for Windows platforms: instead of "L", check the selection box.
		The *transport protocol* to be used (ACI or CSCI). ACI is the default. CSCI is supported only for OpenVMS.
Default setting	none	Subparameter defaults, see above.
Dynamic specification	YES	See below.
Specification within session	YES	At runtime, this value can be overwritten by a Natural user exit (USR2007N).

Syntax for dynamic specification:
DFS=(*server-name*,*server-node name*,*logon-indicator*,*transport-protocol-name*)

RDS - Define Remote Directory Servers

This parameter allows you to define up to 10 remote directory servers. For each remote directory server, you specify up to 5 subparameters.

Possible settings		
	5 subparameters	The *server name* (1 - 8 characters). There is no default, the value must be specified.
		The *server node* (1 - 8 characters). There is no default, the value must be specified.
		The name of the *subprogram*, titled CALLNAT, (1 - 8 characters) to be used as interface (default is RDSSCDIR).
		A *logon indicator*. If nothing is specified, blank is the default. L The client initiates a Natural logon to the server with the library name of the current library on the client. blank No server logon will be executed. Note for Windows platforms: instead of "L", check the selection box.
		The *transport protocol* to be used (ACI or CSCI). ACI is the default. CSCI is supported only for OpenVMS.
Default setting	none	Subparameter defaults, see above.
Dynamic specification	YES	See below.
Specification within session	NO	

For dynamic specification the syntax is as follows.

Using 1 server:
RDS=(*server-name*,*server-node name*,*subprogram*,*logon-indicator*,*transport-protocol-name*)

Using 2-10 servers:
RDS=((*server-name*,*server-node name*,subprogram,*logon-indicator*,*transport-protocol-name*)(*server-name*,*server-node name*,subprogram,*logon-indicator*,*transport-protocol-name*)...(*server-name*,*server-node name*,subprogram,*logon-indicator*,*transport-protocol-name*))

For information on remote directories, see Natural Remote Procedure Call documentation.

TIMEOUT - Wait Time for RPC Server Response

It specifies the number of seconds the client is to wait for an RPC server response. If this time is exceeded, the remote procedure call will be terminated with a corresponding error message.

Possible settings	0 - 32767	Seconds.
Default setting	55	
Dynamic specification	YES	
Specification within session	YES	At runtime, this value can be overwritten by the parameter maintenance of the SYSRPC utility.

 Copyright © Software AG 2002

TRYALT - Try Alternative Server

It determines whether an RPC client should try to execute an RPC request on an alternative server (ON) or not (OFF). For further information, see Specifying RPC Server Adresses in the Natural Remote Procedure Call documentation.

Possible settings	ON	If a request could not be executed on the node you specified, the RPC client tries to find an alternative server address to send that request to.
	OFF	No such attempt will be made.
Default setting	OFF	
Dynamic specification	YES	
Specification within session	YES	At runtime, this value can be overwritten by the parameter maintenance of the SYSRPC utility.

NTSCTAB Macro - Overwriting of Scanner Character-Type Table

The NTSCTAB macro allows you to overwrite the definitions in the scanner character-type table NTSCTAB as contained in the configuration module NATCONFG. The NTSCTAB table defines the properties of characters used in mask definitions for the SCAN/MASK functions.

This document covers the following topics:

- NTSCTAB Syntax
- Example of NTSCTAB Macro
- NTSCTAB Alternative

NTSCTAB Syntax

You specify the NTSCTAB macro as follows:

NTSCTAB *character,attribute-type(s)*

character

You specify a character, and after it its attribute type(s)

You can specify the character either as the one-byte character itself (enclosed in apostrophes) or as the two-byte hexadecimal representation of that character. For each character, you have to specify a separate NTSCTAB macro.

attribute-type(s)

Attribute types can be:

UPPER	upper-case alphabetical
LOWER	lower-case alphabetical
NUM	numeric
HEX	hexadecimal
ALFANUM	alphanumeric
SPECIAL	special
NDELIM	non-delimiter

Example of NTSCTAB Macro

```
NTSCTAB  5E,LOWER,NDELIM
NTSCTAB  'ß',SPECIAL
NTSCTAB  7B,SPECIAL
NTSCTAB  Ä,UPPER,NDELIM
```

 Copyright © Software AG 2002

NTSCTAB Alternative

Instead of the NTSCTAB macro, you can also use the **profile parameter** SCTAB. With the SCTAB parameter, you must enclose the entire string of character/attribute pairs in parentheses. It is possible to specify more than one character in the list of values. Apostrophes must be used for special characters like Ä, ß, etc.

Example:

```
SCTAB=(5E,LOWER,NDELIM,'ß',SPECIAL,7B,SPECIAL,'Ä',UPPER,NDELIM)
```

NTSORT Macro - Control of Sort Program

The NTSORT macro can be used to specify various options that control the handling of the sort program used when a SORT statement is executed.

The sort program to be used can be either Natural's internal one (the default for all environments) or an external one. The type of sort to be used depends on the setting of the NTSORT keyword parameter EXT.

This document covers the following topics:

- NTSORT Syntax
- NTSORT Keyword Parameters
- NTSORT Alternative

NTSORT Syntax

The NTSORT macro is specified as follows:

```
NTSORT
WRKSIZE=nnn,STORAGE=medium,EXT=ON/OFF,EXTNAME=name,EXTOPT=(options,...),EXTEOJ=ON/OFF
```

The individual keyword parameters are explained below.

NTSORT Keyword Parameters

The following keyword parameters are available: WRKSIZE | STORAGE | EXT | EXTNAME | EXTOPT | EXTEOJ

WRKSIZE - Size of Work Buffer Used by Sort Program

WRKSIZE specifies the size *nnn* (in KB) of the work buffer used by the sort program.

Possible values:	10 to 256, or 0.
Default value:	10

If you specify WRKSIZE=0, no sort operations can be performed.

The work buffer specified by WRKSIZE accommodates internal sort control data. The remaining storage is used to collect and sort the records. The size of the sort control data depends on various factors (the WRKSIZE itself, the sort record length, the number of sort keys, their size and format) and can therefore not be calculated in a formal way.

STORAGE - Type of Storage Medium

STORAGE specifies the type of storage *medium* to be used by Natural's internal sort program.

In general, the SORT statement first tries to use the remaining storage in WRKSIZE. If the number of records does exceed this storage, the SORT statement tries to use intermediate storage to additionally process records:

 Copyright © Software AG 2002

MAIN	(default), only the remaining storage of WRKSIZE is used, no other intermediate storage is available.
BP	The SORT buffer pool is used as intermediate storage.
SD	Com-plete SD files are used as intermediate storage.

Note Concerning SD Files

The files are allocated as temporary SD files. They are allocated for a stack level. This means, the name of the SORT SD files are &&STsnnn with:

&&	Indicator for a temporary SD file
ST	Standard prefix for the SD SORT file
s	Stack level
nnn	Sequence number within a single SORT run

EXT - Use of External Sort Program

EXT specifies if an external sort program is to be used or not:

ON	An external sort program will be used. The use of an external sort program is possible only in batch environments, including IMS/BMP, TSO, TIAM and CMS.
OFF	The Natural sort program will be used (this is the default).

EXTNAME - Name of External Sort Program

EXTNAME specifies the *name* (1 to 8 characters) of the external sort program to be used. The default name is "SORT".

EXTOPT - Additional Options for External Sort Program

EXTOPT specifies additional *options* for the external sort program.

Natural generates the necessary field and format parameters and passes them to the external sort program. With EXTOPT, you can specify additional parameters to be passed to the external sort program.

You can only specify parameters that are part of the control statement syntax of your external sort program. Depending on the underlying operating system, the options are appended to the following control statements:

Under OS/390:	SORT control statement
Under VSE/ESA:	OPTION control statement

EXTOPT Example:

The additional parameters can be specified as in the following example:

EXTOPT=(SIZE=E2000000,NOEQUALS,DYNALLOC=(3350,8))

EXTEOJ - Action in the Event of an Error

EXTEOJ specifies the action to be taken if an error is detected during the execution of the external sort program:

ON	If an error is detected, sort processing is terminated. ON requires that the sort program used is able to detect a return code of 16 from both the E15 and E35 sort exit routines.
OFF	If an error is detected, Natural withholds further calls to the sort program and ignores each record as it is passed to the E35 sort exit routine (this is the default).

NTSORT Alternative

All options of the NTSORT macro can also be specified dynamically with the **profile parameter** SORT.

 Copyright © Software AG 2002

NTSYS Macro - Specification of Set of Dynamic Parameters

The NTSYS macro can be used in conjunction with the dynamic parameter SYS to pass a set of dynamic parameters to Natural when it is invoked. The parameters are passed to Natural as dynamic parameters and override the corresponding parameters in the NTPRM macro of the Natural parameter module.

This document covers the following topics:

- NTSYS Syntax
- Examples of NTSYS Macros

For the use of dynamic parameters, see Dynamic Assignment of Parameter Values.

For more information on how to specify a string of dynamic parameters with the PROFILE parameter, see PROFILE.

NTSYS Syntax

An NTSYS macro is specified as follows:

NTSYS *set-name,'parameter=value,...'*

set-name

The *set-name* identifies the subsequent set of parameters, it can be 1 to 8 characters long and must begin with an alphabetical character.

parameter

After the *set-name*, you specify individual profile parameters and their values. For the profile parameters you can specify, see Profile Parameters in the Parameter Reference documentation. The entire set of parameters you specify with an NTSYS macro must constitute a valid string of dynamic parameters.

Passing the Parameter Set to Natural

The set of parameters is passed to Natural by specifying the *set-name* with the dynamic parameter SYS when Natural is invoked:

`SYS=set-name`

Macro Coding Conventions

The following macro coding conventions apply:

- As Assembler macro-coding conventions limit the length of a macro parameter to 255 characters; the parameter string you specify in the NTSYS macro must be split into two or more substrings if the parameters with their values exceed 255 characters.
- Each substring is enclosed in apostrophes, and a comma must be placed after the last parameter specification *within* the first substring to separate it from the first parameter specification in the second

substring.

- Continuation to another line is accomplished by placing a comma *after* the last substring on the current line, inserting a non-blank character in column 72 and continuing with the next substring on the next line starting in position 16.
- An apostrophe within a substring is represented by two apostrophes.

Examples of NTSYS Macros

```
NTSYS SET1,'FUSER=(,50),LC=ON,NC=ON,ULANG=2,TQ=OFF'
NTSYS SET2,'FUSER=(,51),ULANG=4,WH=ON,KC=ON'

NTSYS SET3,'FUSER=(,50),LC=ON,NC=ON,',X'
ULANG=2,TQ=OFF'
```

In the last example, a comma must be placed after NC=ON (and *before* the apostrophe), because otherwise the separating comma between NC=ON and ULANG=2 would be missing.

The comma *after* the apostrophe indicates that the NTSYS macro specification continues on the next line.

 Copyright © Software AG 2002

NTTAB Macro - Overwriting of Standard Output Translation Table

The NTTAB macro allows you to overwrite the definitions in the translation table NTTAB as contained in the configuration module NATCONFG. The NTTAB table is the standard output translation table.

This document covers the following topics:

- NTTAB Syntax
- Example of NTTAB Macro
- NTTAB Alternative

NTTAB Syntax

You specify the NTTAB macro as follows:

```
NTTAB a1,a2,b1,b2,c1,c2,...
```

You specify pairs of characters, the first character of a pair being the character to be translated, the second character of a pair being the character into which the first character is to be translated.

You can specify each character either as the one-byte character itself (enclosed in apostrophes) or as the two-byte hexadecimal representation of that character.

Example of NTTAB Macro

```
NTTAB 5E,'Ä','ö',78,FF,00,'ü','Ü'
```

In this example, the character represented by H'5E' is translated into "Ä", "ö" into the character represented by H'78', the character represented by H'FF' into the character represented by H'00', and "ü" into "Ü".

NTTAB Alternative

Instead of the NTTAB macro, you can also use the dynamic profile parameter TAB.

With the TAB parameter, you must enclose the entire string of character pairs in parentheses, for example:

```
TAB=(5E,'Ä','ö',78,FF,00,'ü','Ü')
```

NTTAB1 Macro - Overwriting of Alternate Output Translation Table

The NTTAB1 macro allows you to overwrite the definitions in the translation table NTTAB1 as contained in the configuration module NATCONFG. The NTTAB1 table is the alternative output translation table for the secondary character set used when the profile/session parameter PM=C is set.

This document covers the following topics:

- NTTAB1 Syntax
- NTTAB1 Alternative

NTTAB1 Syntax

You specify the NTTAB1 macro in the same way as the NTTAB macro; that is you specify pairs of characters, the first character of a pair being the character to be translated, the second character of a pair being the character into which the first character is to be translated.

You can specify each character either as the one-byte character itself (enclosed in apostrophes) or as the two-byte hexadecimal representation of that character.

NTTAB1 Alternative

Instead of the NTTAB1 macro, you can also use the dynamic profile parameter TAB1.

 Copyright © Software AG 2002

NTTAB2 Macro - Overwriting of Alternate Input Translation Table

The NTTAB2 macro allows you to overwrite the definitions in the translation table NTTAB2 as contained in the configuration module NATCONFG. The NTTAB2 table is the alternate input translation table for the secondary character set used when the profile/session parameter PM is set to PM=C.

This document covers the following topics:

- NTTAB2 Syntax
- NTTAB2 Alternative

NTTAB2 Syntax

You specify the NTTAB2 macro in the same way as the NTTAB macro; that is you specify pairs of characters, the first character of a pair being the character to be translated, the second character of a pair being the character into which the first character is to be translated.

You can specify each character either as the one-byte character itself (enclosed in apostrophes) or as the two-byte hexadecimal representation of that character.

NTTAB2 Alternative

Instead of the NTTAB2 macro, you can also use the dynamic profile parameter TAB2.

NTTABA1 Macro - Overwriting of EBCDIC-ASCII Translation Table

The NTTABA1 macro allows you to overwrite the definitions in the translation table NTTABA1 as contained in the configuration module NATCONFG. The NTTABA1 table is used for EBCDIC-to-ASCII translation.

This document covers the following topics:

- NTTABA1 Syntax
- NTTABA1 Alternative

NTTABA1 Syntax

You specify the NTTABA1 macro in the same way as the NTTAB Macro; that is you specify pairs of characters, the first character of a pair being an EBCDIC character to be translated, the second character of a pair being the ASCII character into which the EBCDIC character is to be translated.

You can specify each character either as the one-byte character itself (enclosed in apostrophes) or as the two-byte hexadecimal representation of that character.

NTTABA1 Alternative

Instead of the NTTABA1 macro, you can also use the dynamic profile parameter TABA1.

 Copyright © Software AG 2002

 NTTABA2 Macro - Overwriting of ASCII-EBCDIC Translation Table

NTTABA2 Macro - Overwriting of ASCII-EBCDIC Translation Table

The NTTABA2 macro allows you to overwrite the definitions in the translation table NTTABA2 as contained in the configuration module NATCONFG. The NTTABA2 table is used for ASCII-to-EBCDIC translation.

This document covers the following topics:

- NTTABA2 Syntax
- NTTABA2 Alternative

NTTABA2 Syntax

You specify the NTTABA2 macro in the same way as the NTTAB macro; that is you specify pairs of characters, the first character of a pair being an ASCII character to be translated, the second character of a pair being the EBCDIC character into which the ASCII character is to be translated.

You can specify each character either as the one-byte character itself (enclosed in apostrophes) or as the two-byte hexadecimal representation of that character.

NTTABA2 Alternative

Instead of the NTTABA2 macro, you can also use the dynamic profile parameter TABA2.

Copyright © Software AG 2002

NTTABL Macro - Overwriting of SYS-Library Output Translation Table

The NTTABL macro allows you to overwrite the definitions in the translation table NTTABL as contained in the configuration module NATCONFG. The NTTABL table is used to translate output produced by programs located in "SYS" libraries.

This document covers the following topics:

- NTTABL Syntax
- NTTABL Alternative

NTTABL Syntax

You specify the NTTABL macro in the same way as the NTTAB macro; that is you specify pairs of characters, the first character of a pair being the character to be translated, the second character of a pair being the character into which the first character is to be translated.

You can specify each character either as the one-byte character itself (enclosed in apostrophes) or as the two-byte hexadecimal representation of that character.

NTTABL Alternative

Instead of the NTTABL macro, you can also use the dynamic profile parameter TABL.

 Copyright © Software AG 2002

NTTF Macro - Translation of Database ID / File Number

An NTTF macro is used to translate the database ID / file number of a production database into the database ID / file number of a test database.

This feature is relevant when developing an application in a production environment; it enables you to develop an application in a test database and then transfer the finished application to the production database without having to change or re-compile the application.

The transfer is done by a translation of the application's database identification (DBID) and file number (FNR). The translation is done by using an NTTF macro in the Natural parameter module or with the dynamic profile parameter TF. For each file number to be translated, you must specify one NTTF macro.

In the NTTF macro (or with the TF parameter), you specify the production DBID/FNR and the test DBID/FNR. The Natural objects are then cataloged with the production DBID/FNR, but whenever a database access is executed, the production DBID/FNR is translated into the test DBID/FNR; that is, the test database is used. This means that testing can take place in the actual production environment, but not with production data.

This document covers the following topics:

- NTTF Syntax
- Examples of NTTF Macros

NTTF Syntax

An NTTF macro is specified as follows:

NTTF *production-DBID,production-FNR,test-DBID,test-FNR*

To translate all FNRs in the specified *production-DBID* to the specified *test-DBID* and *test-FNR*

- Specify an asterisk for the *production-FNR*:

  `TF=(production-DBID,*,test-DBID,test-FNR)`

To translate *all* production DBIDs and FNRs to the specified *test-DBID* and *test-FNR*

- Specify asterisks for the *production-DBID* and *production-FNR*:

  `TF=(*,*,test-DBID,test-FNR)`

To transfer the finished application from testing to production

- Simply switch off the DBID/FNR translation by using another Natural nucleus in which the NTTF macro is omitted from the Natural parameter module.

  The programs then use the original DBID/FNR with which they were cataloged; that is, the production DBID/FNR.

Examples of NTTF Macros

```
NTTF 10,21,30,1
  NTTF 10,22,40,2
```

 Copyright © Software AG 2002

NTTRACE Macro - Define Components for Tracing

This macro is intended primarily for Software AG internal use for debugging purposes. It can be used to define the components for which trace data are to be written.

This document covers the following topics:

- NTTRACE Syntax
- Example of NTTRACE Macro
- NTTRACE Alternative

NTTRACE Syntax

An NTTRACE macro is specified as follows:

NTTRACE *trace-ID*,...

trace-ID

As *trace-IDs*, you specify the components to be traced. A *trace-ID* is a character string of up to 8 bytes.

Example of NTTRACE Macro

```
NTTRACE NATGETM,NATFREM
```

This macro causes trace data to be written for the Natural nucleus components NATGETM and NATFREM.

NTTRACE Alternative

Instead of the NTTRACE macro, you can also use the dynamic profile parameter TRACE. With the TRACE parameter, you specify the list of trace IDs enclosed in parentheses, for example:

```
TRACE=(NATGETM,NATFREM)
```

The destinations to which the trace data is written are controlled by the profile parameters ITRACE and ETRACE.

Copyright © Software AG 2002

NTUSER Macro - Use of Alternative Parameter Module

This macro allows you to restrict the use of a parameter module as an alternative parameter module. An alternative parameter module is specified with the dynamic profile parameter PARM.

This document covers the following topics:

- NTUSER Syntax
- Example of NTUSER Macro
- NTUSER Alternative

NTUSER Syntax

You specify the NTUSER macro as follows:

```
NTUSER user-id,...
```

user-id

In the Natural parameter module, you specify an NTUSER macro with the IDs of the users who are to be allowed to use the parameter module as an alternative parameter module. Only the specified users will then be allowed to specify the name of that parameter module with the PARM parameter.

Example of NTUSER Macro

```
NTUSER USER1,USER2,USER3
```

When the PARM parameter is evaluated, Natural checks whether the specified parameter module contains an NTUSER macro.

- If it does, Natural checks whether the current user ID (that is, the current value of the system variable *INIT-USER) is contained in the list of user IDs specified with that NTUSER macro.
- If it is not, the user receives a corresponding error message, and the evaluation of dynamic parameter continues with the next parameter after PARM.
- If the specified parameter module does not contain an NTUSER macro, anyone may specify the module name with a PARM parameter.

NTUSER Alternative

See also the profile parameter USER.

 Copyright © Software AG 2002

 NTUTAB1 Macro - Overwriting of Lower-to-Upper-Case Translation Table

NTUTAB1 Macro - Overwriting of Lower-to-Upper-Case Translation Table

The NTUTAB1 macro allows you to overwrite the definitions in the translation table NTUTAB1 as contained in the configuration module NATCONFG. The NTUTAB1 table is used for lower-to-upper-case translation.

This document covers the following topics:

- NTUTAB1 Syntax
- NTUTAB1 Alternative

NTUTAB1 Syntax

You specify the NTUTAB1 macro in the same way as the NTTAB Macro; that is you specify pairs of characters, the first character of a pair being a lower-case character to be translated, the second character of a pair being the upper-case character into which the lower-case character is to be translated.

You can specify each character either as the one-byte character itself (enclosed in apostrophes) or as the two-byte hexadecimal representation of that character.

NTUTAB1 Alternative

Instead of the NTUTAB1 macro, you can also use the dynamic profile parameter UTAB1.

Copyright © Software AG 2002

 NTUTAB2 Macro - Overwriting of Upper-to-Lower-Case Translation Table

NTUTAB2 Macro - Overwriting of Upper-to-Lower-Case Translation Table

The NTUTAB2 macro allows you to overwrite the definitions in the translation table NTUTAB2 as contained in the configuration module NATCONFG. The NTUTAB2 table is used for upper-to-lower-case translation.

This document covers the following topics:

- NTUTAB2 Syntax
- NTUTAB2 Alternative

NTUTAB2 Syntax

You specify the NTUTAB2 macro in the same way as the NTTAB macro; that is you specify pairs of characters, the first character of a pair being an upper-case character to be translated, the second character of a pair being the lower-case character into which the upper-case character is to be translated.

You can specify each character either as the one-byte character itself (enclosed in apostrophes) or as the two-byte hexadecimal representation of that character.

NTUTAB2 Alternative

Instead of the NTUTAB2 macro, you can also use the dynamic profile parameter UTAB2.

 Copyright © Software AG 2002

NTWORK Macro - Work File Assignments

With NTWORK macros, you define the work files to be used during the session. Within a session, up to 32 logical work files (numbered 1 to 32) can be used.

For an online session, all work files to be used have to be defined with an NTWORK macro (or with the corresponding dynamic profile parameter WORK).

The software components for accessing work files in different environments are called *access methods*. For the duration of a Natural session, each logical work file can be attached to only one access method. The access method for a work file is determined by the keyword parameter AM (see below).

In a batch session, any work files not attached to a specific access method will automatically be detected by the standard batch access method, provided that they have been predefined in the JCL.

Alternatively, OS/390 batch work files need not be predefined in the JCL. They can be defined with the subparameter AM=STD of the NTPRINT macro or PRINT parameter and allocated dynamically during the session by a Natural program using the DEFINE statement or user exit USR2021. See also FAMSTD.

This document covers the following topics:

- NTWORK Syntax
- NTWORK Keyword Parameters for All Environments
- NTWORK Keyword Parameters for AM=STD in All Environments
- NTWORK Keyword Parameters for AM=STD in OS/390 Environments
- NTWORK Keyword Parameters for AM=STD in VSE/ESA Environments
- NTWORK Keyword Parameters for AM=STD in BS2000/OSD Environments
- NTWORK Keyword Parameters for AM=CICS
- NTWORK Keyword Parameters for AM=COMP (Com-plete)
- NTWORK Alternative

NTWORK Syntax

In an NTWORK macro, you first specify one or more logical work file numbers, and then several keyword parameters, which define the characteristics which are to apply to these work files:

NTWORK (*work-file-numbers*),*keyword-parameters*,...

work-file-numbers

The file numbers must be specified first and enclosed in parentheses. The numbers can be from 1 to 31. They can be specified in any sequence. Multiple numbers must be separated from one another by commas or blanks. To specify a range of numbers, you can use a hyphen (-).

keyword-parameters

The keyword parameters for the various environments are described below

- for All Environments,
- for AM=STD in All Environments,
- for AM=STD in OS/390 Environments,

- for AM=STD in VSE/ESA Environments,
- for AM=STD in BS2000/OSD Environments,
- for AM=CICS.
- for AM=COMP (Com-plete),

For work files with different characteristics, you specify different NTWORK macros.

Examples:

```
NTWORK (2,12,18),AM=STD,DEST='WORK**'
  NTWORK (1,3,6-11,15),AM=COMP,OPEN=INITOBJ,CLOSE=CMD
```

NTWORK Keyword Parameters for All Environments

The following keyword parameters are available: AM | DEST | OPEN | CLOSE

AM - Type of Access Method

AM=*xxx* specifies the type of access method to be used:

STD	Standard sequential files (batch, TSO, TIAM, CMS OS simulation).
COMP	Com-plete print files.
CICS	CICS transient data or temporary storage.
CMS	CMS Disk and SFS files.
PC	Entire Connection.
USER	Third-party vendor work-file interface.
OFF	Unassigned.

Note:
WORK=OFF is equivalent to: WORK=((1-32)), AM=OFF).
It does not affect any of the other keyword specifications.

DEST - External Dataset Name

DEST=*name* specifies the external dataset name (1 - 8 characters).

The meaning of this keyword parameter depends on the access method. For AM=STD, it is the logical dataset name (DDNAME).

If the destination is to be for multiple files, two asterisks (**) have to be specified for the file number. These will then be replaced by the corresponding logical file number for each work file.

The default value of DEST depends on the access method. For AM=STD, the default is DEST='CMWKF**' (note that a DEST value including two asterisks must be enclosed in apostrophes when using it as a dynamic parameter).

 Copyright © Software AG 2002

Under CICS:	There is no DEST default value for print files under CICS. Here, the DEST parameter is mandatory; that is, CICS work files without a valid DEST specification are ignored.
Under CMS:	For usage of DEST under CMS, refer to Natural under VM/CMS (in the Natural Operations documentation for mainframes).
Under VSE/ESA:	Only 7-character names are supported.

OPEN - Time of File Opening

OPEN=*xxx* determines when the file is to be opened:

Option	The file is opened
INIT	for output at session initialization.
OBF	sets the default OPEN value for the different environments (Batch, CICS, Com-plete, TSO).
OBJ	when the execution of the first object which accesses the file starts (this is the default).
INITOBF	for output at session initialization. Any subsequent re-opening of the file sets the default OPEN value for the different environments (Batch, CICS, Com-plete, TSO).
OBJ1	when the execution of the first object on level 1 which accesses the file starts. Otherwise, it is opened when it is first accessed.
ACC	when it is first accessed by a statement.
INITOBJ	for output at session initialization. Any subsequent re-opening of the file will be performed when the execution of the first object which accesses the file starts.
INITOBJ1	when the execution of the first object on level 1 which accesses the file starts. Otherwise, it is opened when it is first accessed.
INITACC	for output at session initialization. Any subsequent re-opening of the file will be performed when it is first accessed by a statement.

CLOSE - Time of File Closure

CLOSE=*xxx* determines when the file is to be closed:

Option	The file is closed
OBJ	either when processing of the object in which it was first accessed is finished, or when command mode, NEXT mode or MAINMENU is reached.
CMD	when command mode, NEXT mode or MAINMENU is reached (this is the default).
FIN	at session end. With CLOSE=FIN, a CLOSE WORK FILE statement does not cause the work file to be closed. When the end-of-file condition occurs during the READ WORK FILE statement, Natural closes the work file immediately.
CLOSE=USER	This value specifies that a work file is closed only if the file is open and one of the following conditions is true: • a CLOSE WORK statement is issued, • a DEFINE WORK statement is issued, • at session termination.

NTWORK Keyword Parameters for AM=STD in All Environments

The following keyword parameters are available: RECFM | BLKSIZE | LRECL | TRUNC | PAD

RECFM - Default Record Format of Dataset

RECFM=*xxxx d*etermines the default record format of the dataset.

The following formats are supported:

F	Fixed
V	Variable
U	Undefined
B	Blocked
S	Spanned
A	ASA
M	Machine control characters

The following values and also combinations of values are possible:

Possible value:	F, FA, FM, FB, FBA, FBM, V, VA, VM, VB, VBA, VBM, VBS, VBSA, VBSM, U, UA, UM.
Default value:	RECFM=VB (variable blocked).

The RECFM specification only applies if no record format is predefined in the JCL or in the dataset DCB (OS/390 only).

BLKSIZE - Default Block Size of Dataset

BLKSIZE=*nnnnn* determines the default block size (in bytes) of the dataset.

Possible values:	0, or 8 to 32767.
Default value:	4628

The BLKSIZE specification only applies if no block size is predefined in the JCL or in the dataset DCB (OS/390 only).

LRECL - Default Record Length of Dataset

LRECL=*nnn d*etermines the default record length (in bytes) of the dataset.

Possible values:	0, or 5 - 32767.
Default value:	0

 Copyright © Software AG 2002

This parameter is used particularly to check for truncation and padding.

- For RECFM=V (B) the LRECL value includes a 4-byte record descriptor word.
- If LRECL = 0 is defined, the following applies:
  - With RECFM=V (B), LRECL defaults to BLKSIZE-4.
  - With RECFM=U, LRECL defaults to BLKSIZE.
  - With RECFM=F (B), the maximum record length in the Natural program being executed is taken when the file is opened. If no record length from a program is available when the file is opened, for example with OPEN=INIT, this leads to an error.

The LRECL specification only applies if no record length is predefined in the JCL or in the dataset DCB (OS/390 only).

TRUNC - Truncation of Output Records

TRUNC=ON/OFF determines whether the output records are truncated or not:

ON	Output records that are longer than the record length (LRECL) of the dataset will be truncated.
OFF	Error NAT1512 will be issued if an output record is longer than the dataset record length (this is the default).

PAD - Padding of output records

PAD=ON/OFF determines whether the output records are padded with blanks or not (applies only to datasets of fixed record length):

ON	Output records that are shorter than the record length (LRECL) of the dataset will be padded with hexadecimal zeros (this is the default).
OFF	Error NAT1510 will be issued if an output record is shorter than the dataset record length.

NTWORK Keyword Parameters for AM=STD in OS/390 Environments

The following keyword parameters are available: REREAD | FREE | BUFNO | DISP | VMAX

REREAD - Closing of Tape File

REREAD=*xxx* sets the REREAD option for the closing of the tape file:

ON	The REREAD option is set for the CLOSE SVC. This causes the volume to be repositioned to reprocess the dataset (this is the default).
OFF	The REREAD option is not set for the CLOSE SVC.

FREE - Dataset De-allocation at File Closure

FREE=*xxx* determines whether the dataset is de-allocated when the file is closed:

ON	The FREE option is set for the CLOSE SVC, which means that the dataset is de-allocated when it is closed (and not at step termination).
OFF	The FREE option is not set for the CLOSE SVC (this is the default).

BUFNO - Default Number of OS/390 I/O Buffers of Dataset

BUFNO=*nnn* defines the default number of OS/390 I/O buffers of the dataset.

Possible values	0 - 255.
Default value	0. In this case, OS/390 allocates five I/O buffers per default.

The number of I/O buffers can improve the performance of print and work file access dramatically. Note that the storage for I/O buffers is allocated below the 16 MB line.

Note:
The BUFNO specification applies only if no BUFNO value is predefined in the JCL.

DISP - Open Work File for Modification

DISP=MOD determines that the print file is opened for modification, i.e. the new records are added at the end of the file.

This corresponds to the JCL DD statement subparameter DISP=MOD.

VMAX - Enforce LRECL=BLKSIZE-4 for Variable Record Format

VMAX=*xxx* enforces LRECL=BLKSIZE-4 for variable record format.

ON	Providing a nonzero BLKSIZE value exists for the file, VMAX=ON enforces LRECL=BLKSIZE-4 for variable record format, regardless of the LRECL setting in the DCB or the LRECL subparameter.
OFF	Default value.

NTWORK Keyword Parameters for AM=STD in VSE/ESA Environments

The following keyword parameters are available: SYSNR | LABEL | REWIND

SYSNR - Logical VSE SYS Number

SYSNR=*nn* determines the logical VSE SYS number.

Possible values:	1 - 99.
Default value:	By default, the SYS number is identical to the work file number.

LABEL - Tape Label Processing

LABEL=*xxx d*etermines the tape label processing:

 Copyright © Software AG 2002

ON	The tape is in standard label format (this is the default).
OFF	The tape is unlabeled with front tape mark.
NOTM	The tape is unlabeled without front tape mark.

REWIND - Action at File Closure

REWIND=*xxx* determines the action to be taken when a tape file is closed:

ON	The tape is rewound when the file is closed (this is the default).
OFF	The tape is not rewound when the file is closed.
UNLOAD	The tape is unloaded when the file is closed.

NTWORK Keyword Parameters for AM=STD in BS2000/OSD Environments

The following keyword parameter is available: DISP

DISP - File Open Mode

DISP=*xxx d*etermines the open mode of the file:

EXT	The open mode is set to EXTEND.
All user values	The open mode is set to the default value OUTPUT.

NTWORK Keyword Parameters for AM=CICS

The following keyword parameters are available: TYPE | DISP

TYPE - Type of CICS Storage Medium

TYPE=*xxxx* specifies the type of CICS storage medium to be used:

MAIN	Temporary main storage.
AUX	Temporary auxiliary storage.
TD	Transient data.

DISP - CICS Storage Queue Disposition

DISP=(*xxx*,*xxx*) specifies the CICS storage queue disposition.

Possible value pairs are:

(NEW,KEEP)	The storage queue is deleted when the file is opened (this is the default)
(NEW,DELETE)	The storage queue is deleted when the file is opened and when it is closed.
(OLD,DELETE)	The storage queue is deleted when the file is closed.
(OLD,KEEP)	The storage queue is not deleted.

Note:
The DISP specification does not apply to CICS extra-partition transient data queues.

NTWORK Keyword Parameters for AM=COMP

The following keyword parameters are available for AM=COMP (Com-plete): TYPE | BLOCKS | BLKSIZE

TYPE - Type of Storage Access

TYPE=*xxx* specifies the type of storage access to be used:

SHR	Shared access, that is, the work file is accessible by all users.
TID	The work file is only available to the current Com-plete terminal ID.
DYN	The work file is only available to the current terminal stack level.

BLOCKS - Number of Storage Blocks

BLOCKS=*nnnn* specifies the number of storage blocks to be allocated.

Possible values:	1 to 9999
Default value:	20

BLKSIZE - Size of Storage Blocks

BLKSIZE=*nnnn* determines the default block size (in bytes) of the dataset.

Possible values:	0, or 8 - 32767
Default value:	4628

NTWORK Alternative

Instead of the NTWORK macro, you can also use the dynamic profile parameter WORK. With the WORK parameter, you have to specify the entire string of values enclosed in parentheses. For example:

```
WORK=((1,2,5-12,18),AM=STD,OPEN=INITOBJ,CLOSE=OBJ)
```

 Copyright © Software AG 2002

Command Reference - Overview

This document describes the **Natural system commands**. The following topics are covered:

- What are System Commands?
  Introduction to Natural system commands.
- Issuing System Commands
  How to issue system commands on different platforms
- Command Syntax
  Information on the syntax symbols used in this section.
- Commands Grouped by Functions
  System commands grouped according to their functions.
- Command List
  Detailed explanation of each system command, in alphabetical order.

What are System Commands?

Natural system commands are part of the Natural runtime system and perform certain operating functions during a Natural session. Natural system commands are used to, for example:

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

System commands provide a key-based alternative to the menu-based means of issuing commands.

Single Point of Development: If you are using a remote development server, commands concerning the remote environment are issued under Windows, but both obtain information from and return information to the mainframe as required. If the command requires input or output from a mainframe screen, a terminal emulation window displaying the appropriate screen appears. Commands are executed locally wherever possible.

Note:
Natural system commands perform functions in a Natural session whereas Natural statements are components of Natural programs.

Issuing System Commands

To issue a system command, do one of the following:

- Enter it in the command line;
- Enter it at the Natural NEXT or MORE prompt.

Commands are only available in an appropriate context.

Command input is not case-sensitive.

Note:
Some Natural commands effect objects other than the currently active object.

Command Line

The Command Line Window in Natural for Mainframes

You can enter commands in the command line at the command prompt.

The Command Line Window in Natural for Windows

You can enter commands in the Command field in the Command Line window. Initially, the Command Line window is not shown.

Note: Under Windows, the functinaity of system commands is also available via various menus

To toggle command line display

From the View menu, choose Command Line.
Or press ALT+3.

When the Command Line window is displayed, a check mark is shown next to the Command Line command.

The Direct Function in Natural for UNIX/Open VMS

If you select the Direct function from the Main Menu, the Direct Command window is displayed in which you enter the name of the system command or program you wish to be executed (provided the program is in your current library, the steplib or the library SYSTEM).

MORE Prompt

You can also enter a system command at a MORE prompt during the execution of a program. A MORE prompt is displayed at the bottom of an output screen to signal when more output is pending. When a system command is entered in response to a MORE prompt, program execution is interrupted and the system command is executed.

 Copyright © Software AG 2002

Syntax

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

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

Example of Command Syntax:

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

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

 Copyright © Software AG 2002

Commands Grouped by Functions

This section provides an overview of the Natural system commands grouped according to their functions:

- Navigating within Natural
- Environment Settings
- Editing and Storing Programming Objects
- Executing Programs
- Invoking Maintenance Utilities
- Invoking Transfer Utilities
- Monitoring and Debugging
- Commands used with NaturalX
- Commands used with SQL Databases
- Commands used with the Natural Optimizer Compiler
- Miscellaneous

Note:
Platform-specific availability information:
M=Mainframes, **U**=UNIX, **O**=OpenVMS, **W**=Windows

Navigating within Natural

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

Environment Settings

Command	Platform	Function
COMPOPT	M,U,O,W	Sets various compilation options that affect the way in which Natural programming objects are compiled.
GLOBALS	M,U,O,W	Changes the settings of various Natural session parameters.
KEY	M,U,O	Assigns functions to keys to be used in your Natural session.
SYSPARM (*)	M	Changes the settings of various Natural profile parameters.
MAP	W	Establishes a connection to a remote development server .
PROFILE	M,U,O,W	Only available if Natural Security has been installed. Displays the security profile currently in effect. This profile informs you of the conditions of use in effect for you in your current Natural environment.
SYSPROD	M,U,O,W	Displays a list of the products installed at your site, and some information on these products.
SYSPROF	M,U,O,W	Displays the current definitions of the Natural system files.
UNMAP	W	Disconnects the currently active remote environment.

 Copyright © Software AG 2002

Editing and Storing Programming Objects

Command	Platform	Function
CATALL	M,U,O,W	Stores **all** objects in the current library in source and/or object form.
CATALOG	M,U,O,W	Compiles the programming object currently in the source work area of an editor and stores the resulting object module in the Natural system file.
CHECK	M,U,O,W	Checks if the source code of a programming object contains any syntax errors. The checking process varies according to the type of object being checked. Syntax checking is also performed as part of the system commands RUN and STOW.
CLEAR	M,U,O,W	Clears the contents of the work area of the editor.
DELETE	M	Deletes a programming object from the Natural system file. You can delete it in source form, in object form, or both.
EDIT	M,U,O,W	Edits the source form of a programming object.
LIST	M,U,O,W	Lists one or more objects which are contained in the current library.
READ	M,U,O,W	Transfers an object in source form from the Natural system file to the source work area.
RENAME	M	Gives a Natural programming object another name.
RENUMBER	M,U,O,W	Renumbers the source code currently in the source work area.
SAVE	M,U,O,W	Saves the **source form** of the programming object currently in the work area of the editor and stores it in the Natural system file.
SCAN	M,U,O,W	Searches for a string of characters within an object, with an option to replace the string with another string.
STOW	M,U,O,W	Saves the **source form** of a programming object, compiles the object and stores the resulting **object module** as well as the source in the Natural system file.
STRUCT	M,U,O	Performs structural indentation of a source program, and helps detecting structural inconsistencies.

Copyright © Software AG 2002

Executing Programs

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

Invoking Maintenance Utilities

Command	Platform	Function
SYSDDM (*)	M,U,O	Creates and maintains Natural data definition modules (DDMs).
SYSERR (*)	M,U,O,W	Creates and maintains the messages you wish your Natural applications to display to the users.
SYSNCP (*)	M,U,O,W	Creates and maintains the command processors to be used in your Natural applications.
SYSRPC (*)	M,U,O,W	Creates and maintains remote procedure calls, that is, provides the settings necessary to execute a subprogram located on a remote server.

Invoking Transfer Utilities

Command	Platform	Function
NATLOAD (*)	M,U,O	Load Natural programming objects, error messages and DDMs from a work file into system files. See also SYSUNLD.
NATUNLD (*)	M,U,O	Unloads Natural programming objects, error messages and DDMs from system files onto a work file. See also SYSUNLD.
SYSMAIN (*)	M,U,O,W	Transfers objects within the Natural system from one library to another.
SYSOBJH	U,O,W	Processes Natural and non-Natural objects for distribution in Natural environments.
SYSTRANS (*)	M,U,O,W	Transfers all Natural programming objects, DDMs, libraries, command processors and error messages as well as Adabas FDTs from one hardware platform to another.
SYSUNLD (*)	M,U,O	Invokes the SYSUNLD utility, which contains the two utilities NATUNLD and NATLOAD.

 Copyright © Software AG 2002

Monitoring and Debugging

Command	Platform	Function
BUS (*)	M	Invokes the Buffer Usage Statistics function of the SYSTP utility. BUS provides statistical information on the usage of Natural buffers.
DUMP	M	Provides information for Software AG technical support personnel in order to locate an error that caused an abnormal termination (abend) of the Natural system.
RPCERR (*)	M	Displays the last Natural error number and message if related to Remote Procedure Call (RPC), and the last BROKER reason code and associated message.
SYSADA (ADACALL) (*)	M	Invokes the ADACALL utility which is contained in the library SYSADA. ADACALL enables you to issue Adabas direct calls (native commands) directly to an Adabas database.
SYSBPM (*)	M	Invokes the SYSBPM utility, which is used to monitor the buffer pool and adjust it to meet your requirements.
SYSEDT (*)	M	Invokes the Editor Buffer Pool Services utility SYSEDT, which is used to display runtime information of the editor buffer pool, modify its parameters, and delete logical work and recovery files.
SYSTP (*)	M	Invokes the SYSTP utility, which allows you to monitor and control various TP-monitor-specific characteristics of Natural.
TECH	M,U,O,W	Displays technical and other information on your Natural session.
TEST (*)	M	Invokes the Natural Debugger for online testing and debugging. The Natural Debugger allows you to test various aspects of your applications and locate errors in the processing flow of programs.

Commands used with NaturalX

Command	Platform	Function
DCOMPARM (*)	M	Displays and modifies the Natural parameters for a specified server ID in the system registry.
REGISTER (*)	M,U,O,W	Registers Natural classes. They are registered for the server ID under which Natural was started.
UNREGISTER (*)	M,U,O,W	Unregisters Natural classes.

Commands used with SQL Databases

Command	Platform	Function
LISTSQL	M	Only available with Natural for DB2 and Natural for SQL/DS. Lists those Natural statements in a programming object which are associated with a database access, and the SQL commands they have been translated into.
SQLERR	M	Only available with Natural for DB2 and Natural for SQL/DS. Displays information on an SQL error.

Copyright © Software AG 2002

Commands used with the Natural Optimizer Compiler

Command	Platform	Function
NOCOPT (*)	M	Displays the current settings of the Natural Optimizer Compiler options as they were specified during Natural startup, and allows you to change these settings.
NOCSHOW (*)	M	Provides buffer information on the output generated by the PGEN option. PGEN causes the Natural Optimizer Compiler to output generated code and internal Natural structures.
NOCSTAT (*)	M	Provides statistical data on programs suitable for processing by the Natural Optimizer Compiler.

Miscellaneous

Command	Platform	Function
AIV	M	Displays all application-independent variables (AIVs) which have been defined.
HELP	M,U,O,W	Invokes the Natural help system.
INPL (*)	M,U,O,W	Invokes the INPL utility. It is **only** used for the loading of Software AG installation datasets into the system files.
LAST	M,U,O,W	Displays the system commands that were last executed, and allows you to execute them again.
LASTMSG	M,U,O,W	Displays additional information on the error situation which occurred last.
LIST COUNT	M,U,O,W	Lists the number of Natural objects in your current library.
MAIL	M,U,O,W	Only available if Natural Security has been installed. Invokes a mailbox to modify its contents and/or expiration date. A mailbox is used as a notice board to broadcast messages to Natural users.
ROUTINES	M	Shows you which objects in the current library use which external subroutines.
SYSEXT	M,U,O,W	Invokes the library SYSEXT, which contains various Natural user exits.
SYSFILE (*)	M,U,O,W	Invokes the function Natural Print/Work Files of the SYSTP utility. This function provides information on the available work files and print files.
UNLOCK	W	Enables you to view locked objects and unlock them if required.
UPDATE	M,U,O,W	Prevents database updating being carried out by a program.
XREF	M,U,O,W	Only available if Predict has been installed. Controls the usage of the Predict function "active cross-references". This function automatically creates documentation in Predict about the objects which a program/data area references.

* Each of these commands invokes a utility or application of a corresponding name. For a description of the utilities, see the Natural Utilities for Mainframes documentation, refer to the list of utilities on the overview page of Natural for Mainframes or see below:

 Copyright © Software AG 2002

DCOMPARM	NaturalX documentation
NOCOPT	Natural Optimizer Compiler documentation
NOCSHOW	Natural Optimizer Compiler documentation
NOCSTAT	Natural Optimizer Compiler documentation
SYSRPC	Natural RPC documentation
REGISTER	NaturalX documentation
UNREGISTER	NaturalX documentation

System Command List

This section provides a detailed explanation of each system command, its usage, parameters and syntax. The commands are listed in alphabetical order.

Note:
Natural system commands are used to perform session functions, whereas Natural statements are components of Natural programs.

See also:

- Syntax
- Issuing Commands

 Copyright © Software AG 2002

AIV

Note:
This command is for Mainframes only.

AIV

With the system command AIV, you can view all application-independent variables (AIVs) which are currently active.

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

What are the F and Len columns for???

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

For information on the definition of AIVs, see the DEFINE DATA statement as described in the Natural Statements documentation and the section User-Defined Variables in the Natural Reference documentation.

Back to System Commands - Overview.

BUS

Note:
This command is for Mainframes only.

This command is used to invoke the function Buffer Usage Statistics of the SYSTP utility. The SYSTP Utility documentation is located in the Debugging and Monitoring section of the Utilities for Mainframes documentation.

This function provides information on the buffers allocated for the current Natural session, their sizes, and the actual buffer space being used.

Back to System Commands - Overview.

 Copyright © Software AG 2002

CATALL

- For Mainframes
- For UNIX/OpenVMS
- For Windows

CATALL for Mainframes

```
CATALL [ object-name [TO object-name] [ RECAT | ALL ] [TYPES types] [ SAVE | CATALOG | STOW | CHECK ] [options...] ]
                                      text-name
```

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

When you enter the CATALL command without any additional options, the Catalog Objects in Library screen is displayed, which you use to perform the following functions:

- Catalog Objects from/to
- Recatalog Only Existing Modules, or Catalog All Sources
- Select Object Types
- Select Function
- Select Options
- Invoke Selection List

You can also issue the CATALL command directly using the command syntax shown above.

- Direct Command Syntax

Catalog Objects from/to

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

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

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

Instead of entering settings in these fields, you can also select objects from a selection list (see below).

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

Recatalog Only Existing Modules, or Catalog All Sources

This option only applies to the functions Catalog and Stow:

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

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

Select Object Types

By default, CATALL applies to objects of all types in the current library (all object types are marked with **X**). If you wish objects of a certain type not to be processed by CATALL, overwrite the respective **X** with a blank.

Select Function

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

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

Select Options

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

Condition Code in Batch	If you execute CATALL in batch mode and mark this option with a character, Condition Code 55 will be returned either if a syntax error is detected during CATALL execution or if no objects are found within the specified range of objects to be processed (applies to CATALOG and STOW only).
Renumber Source-Code Lines	By default, the source-code lines of sources that were Saved or Stowed are also renumbered. If you wish no automatic renumbering of lines, overwrite the **X** in this field with a blank.
Keep Result List	CATALL generates a result list. If you wish to keep this list for further use, mark this field with a character. The library SYSEXT contains the user exit USR1024 which you can use to output the result list.
Processing Information	During online processing, CATALL shows a scrolling display of processing status information. To suppress this display, overwrite the **X** in this field with a blank.
Error Report	At the end of processing, CATALL displays a list of the errors that occurred. To suppress this error list, overwrite the **X** in this field with a blank.

 Copyright © Software AG 2002

Selection List

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

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

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

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

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

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

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

CATALL will then begin to process the selected objects.

Direct Command Syntax

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

object-name TO *object-name*	Corresponds to the fields "Catalog Objects from" and "to" of the CATALL screen.
RECAT / ALL	Corresponds to the options Recatalog Only Existing Modules, or Catalog All Sources of the CATALL screen.

TYPES *types*	Corresponds to the object types marked on the CATALL screen. Possible *types* are: G Global data areas A Parameter data areas L Local data areas C Copycodes T Texts S Subroutines N Subprograms H Helproutines M Maps P Programs 4 Classes * All types (this is the default) The *types* have to be specified as **one** character string, for example, LAG for local, parameter and global data areas.
SAVE / CATALOG / STOW / CHECK	Corresponds to the functions of the same names on the CATALL screen.
options	These options correspond to Select Options on the CATALL screen. Possible *options* are: CC Condition Code will be returned. NOREN No automatic renumbering of source-code lines. KEEP Result list will be kept. NOSCROLL Scrolling display of processing status information will be suppressed. NOREPORT Error report will be suppressed. **Note:** If you specify NOREPORT **and** NOSCROLL, KEEP will automatically apply, too.
text-name	Corresponds to specifying a text name in the "Catalog Objects from" field of the CATALL screen.

Example of CATALL Command:

 Copyright © Software AG 2002

CATALL * STOW KEEP CC NOREN

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

Back to System Commands - Overview.

CATALL for UNIX/OpenVMS

CATALL [*object-name*]

This command is used to catalog all objects in the current library.

The CATALL command invokes the Start Catall window:

```
+--------------------- Start Catall ----------------------+
| Object Name: *                                           |
|                                                          |
| Recatalog only if module exists                    X     |
| Catalog all source objects                               |
|                                                          |
| DDMs                     X  Helproutines           X     |
| Global Data Areas        X  Maps                   X     |
| Local Data Areas         X  Programs               X     |
| Parameter Data Areas     X  Generate new map source      |
| External Subroutines     X  Classes                X     |
| Subprograms              X                               |
|                                                          |
+----------------------------------------------------------+
```

In this window, you can make the following specifications:

- **Object name**
  If you wish CATALL to be performed for all objects of the selected types in the current library, specify "*" as object name.
  If you wish it to be performed for a certain range of objects, you can use asterisk notation for the name.
- **Recatalog only if module exists**
  If you mark this field, only those objects for which object modules already exist in the current library will be cataloged again; objects which only exist in source form will not be cataloged; or:
- **Catalog all source objects**
  If you mark this field, *all* selected objects will be cataloged.
- **Object types**
  By default, CATALL applies to objects of all types in the current library (all object types are marked with "X"). If you wish objects of a certain type *not* to be affected by CATALL, overwrite the respective "X" with a blank.
- **Generate new map source**
  Maps created with previous Natural versions are not necessarily compatible with Natural Version 3.1 and above. Mark this field to ensure that maps are converted during the catalog operation. This option converts and stores maps in source *and* object form.

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

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

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

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

 Copyright © Software AG 2002

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

Back to System Commands - Overview.

CATALL for Windows

```
CATALL [object-name]
```

This command is used to catalog all objects or selected objects in the current library. When you issue this command, the "Catalog selected Objects" / Catalog Objects in Library: name dialog box appears. In this box, you specify which types of objects are to be cataloged. Objects are cataloged in the order in which the object types are listed in the "Catalog selected Objects" dialog box, beginning with DDMs and ending with dialogs.

Catalog Object

Enter "*" for all objects of the selected types in the current library.

Asterisk notation can also be used to further restrict the number of objects to be cataloged.

Catalog all source objects

If you select this action, *all* selected objects will be cataloged.

Catalog only objects in both source and cataloged form

If you select this action, only those objects for which object modules already exist in the current library will be cataloged again; objects that exist only in source form will not be cataloged.

Object Types

By default, CATALL applies to objects of all types in the current library (all object types are selected). If you wish objects of a certain type *not* to be affected by CATALL, deselect it.

Generate New Map Source

Maps created with previous Natural versions are not necessarily compatible with Natural Version 3.1. Mark this box to ensure that maps are converted during the catalog operation. This option converts and stores maps in source *and* object form.

Processing

After the CATALL command has been issued, a statistics window displays in real time the number of objects selected, the number of objects already processed, the number of objects cataloged successfully and the number of objects that could not be cataloged.

After cataloging is completed, choose OK. If any objects were not cataloged successfully, they are displayed in a window in the order they were processed. Associated with each object name is an error number (reason why the object could not be cataloged) and an error line (the line position in source code where the error can be found).

To view an explanation of the error

1. Select the object from the list.
2. Choose OK to edit the object.

Note:
To interrupt a CATALL in progress, press the STOP button.

 Copyright © Software AG 2002

Back to System Commands - Overview.

CATALOG

- For Mainframes
- For UNIX/OpenVMS
- For Windows

CATALOG for Mainframes

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

The CATALOG command is used to compile the Natural programming object currently in the source work area of an editor and store the resulting object module in the Natural system file.

object-name

As *object-name*, you specify the name under which the object is to be cataloged.

If you do not specify an *object-name*, the object is cataloged under the name of the object last read into the source work area (for example, EDIT, READ, RUN).

library-id

If you do not specify a *library-id*, the object is cataloged in the current library. If you want the object to be cataloged into another library, you must specify the *library-id* of that library.

Under Natural Security, you cannot specify a *library-id*; you can only store an object in your current library.

Note:
The CATALOG command cannot be used if the **profile parameter** RECAT has been set to ON; in this case, use the STOW command to ensure that source code and object code match.

For LDAs, GDAs and Maps the CATALOG command cannot be invoked from the Next prompt.

Back to System Commands - Overview.

 Copyright © Software AG 2002

CATALOG for UNIX/OpenVMS

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

The CATALOG command is used to compile the Natural programming object currently in the source work area of an editor and to store the resulting object module.

object-name

As *object-name*, you specify the name under which the object is to be cataloged.

If you do not specify an *object-name*, the object will be cataloged under the name as set by the last command which caused a source object to be read into the editor (for example, EDIT, READ, RUN).

library-id

If you do not specify a *library-id*, the object will be cataloged in the current library. If you want the object to be cataloged into another library, you must specify the *library-id* of that library.

Under Natural Security, you cannot specify a *library-id*; that is, you can store an object only in your current library.

Note:
The CATALOG command cannot be used if the **profile parameter** RECAT has been set to ON; in this case, use the STOW command to ensure that source code and object code match.

Back to System Commands - Overview.

CATALOG for Windows

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

The CATALOG command is used to compile a Natural programming object and store the resulting object module.

The following methods can be used to select the object to be cataloged:

- In the tree view, select the object and choose "Catalog" from the context menu.
- In the tree view, select the object and click the "Catalog" toolbar button.
- From the corresponding editor, by clicking the "Catalog" toolbar button.

Note:
The CATALOG command may have been prohibited by the Natural Security administrator.

Note:
The CATALOG command cannot be used if the **profile parameter** RECAT has been set to ON; in this case, use the STOW command to compile the object.

object-name

To catalog an object under a name other than the source name, specify the *object-name* with the CATALOG command.

Compare this with mainfr and unix.....

Object names within a library must be unique.

library-id

When you catalog an object under a different name, the object will, by default, be stored in the current library. If you wish to store it in another library, specify the desired *library-id* after the *object-name*.

Under Natural Security, you cannot specify a *library-id* because you can only store an object in your current library.

Back to System Commands - Overview.

 Copyright © Software AG 2002

CHECK

- For Mainframes
- For UNIX/OpenVMS and Windows

CHECK for Mainframes

```
CHECK
```

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

In online mode, syntax checking stops when a syntax error is detected, and the source-code line that contains the error is marked with **E** in the program editor.

In batch mode, syntax checking continues until all statements have been checked, and all errors detected are included in the output listing.

Note:
For LDAs, GDAs and Maps the CHECK command cannot be invoked from the Next prompt.

Back to System Commands - Overview.

CHECK for UNIX/OpenVMS and Windows

```
CHECK
```

The CHECK command is used to verify the syntax of a source program.

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

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

Back to System Commands - Overview.

 Copyright © Software AG 2002

CLEAR

- For Mainframes
- For UNIX/OpenVMS and Windows

CLEAR for Mainframes

```
CLEAR
```

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

If this object has been modified since the last save, you are prompted to save any changes.

Back to System Commands - Overview.

CLEAR for UNIX/OpenVMS and Windows

```
CLEAR
```

The CLEAR command closes the currently active object and opens a new editor window without content and without a name. The type of editor is the same as the currently active object. If this object has been modified since the last save, you are prompted to save any changes.

Back to System Commands - Overview.

 Copyright © Software AG 2002

COMPOPT

- For Mainframes
- For UNIX/OpenVMS
- For Windows

COMPOPT for Mainframes

COMPOPT [*option=value*]

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

option=value

The keywords for the individual options are shown on the Compilation Options screen and are described in the section Options.

The settings assigned to a compiler option are in effect until you issue the next LOGON command to another library. At LOGON, the default settings set with the macro NTCMPO and/or profile parameterCMPO will be resumed.

This section covers the following topics:

- General Information on Compiler Options
- Options

General Information on Compiler Options

You can specify compiler parameters on different levels:

- The default settings of the individual parameters are set with the macro NTCMPO in the Natural parameter module NATPARM.
- At session start, you can override the compiler parameters with the profile parameter CMPO.
- During an active Natural session, there are two ways to change the compiler parameters with the COMPOPT command: either directly using command assignment (COMPOPT *option=value*) or by issuing the COMPOPT command without parameters which displays the Compilation Options screen. For further in formation, see the section Options. The settings assigned to a compiler option are in effect until you issue the next LOGON command to another library. At LOGON, the default settings set with the macroNTCMPO and/or profile parameter CMPO (see above) will be resumed.
- In a Natural programming object (for example: program, subprogram), you can set compiler parameters with the OPTIONS statement.
  Example:

```
0010 OPTIONS KCHECK=ON
0020 WRITE 'Hello World'
0030 END
```

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

Options

If you issue the COMPOPT command without parameters, the Compilations Options screen appears.

- KCHECK - Keyword Checking
- PCHECK - Parameter Checking for CALLNAT Statements
- DBSHORT - Interpretation of Database Short Field Names
- PSIGNF - Internal Representation of Positive Sign of Packed Numbers
- TSENABL - Applicability of TS Profile Parameter
- GFID - Generation of Global Format IDs
- LOWSRCE - Allow Lower-Case Source
- FINDMUN - Detect Inconsistent Comparison Logic in FIND Statements
- V22COMP - Compatibility Option for Inconsistent Version 2.2 Syntax

KCHECK - Keyword Checking

ON	Programming objects will be checked for Natural statement keywords. Variable names which are reserved Natural keywords are rejected. The section Keywords and Reserved Words in the Natural Reference documentation contains a list of all Natural keywords and reserved words, in which the statement keywords affected by this option are marked.
OFF	No keyword check is performed. This is the default.

PCHECK - Parameter Checking for CALLNAT Statements

ON	It will be checked if the number of the parameters specified in a CALLNAT statement corresponds with the number of parameters in the subprogram to be invoked - provided that subprogram already exists (if the subprogram does not exist, this option has no effect).
OFF	No parameter check is performed. This is the default.

DBSHORT - Interpretation of Database Short Field Names

ON	Database field names in programming objects are considered long names (as defined in the corresponding DDM) - except 2-character field names, which are considered short names (as used by the underlying database system). This is the default. **Note:** Even if DBSHORT=ON, the use of a short database field is not allowed in the following cases: • if a DEFINE DATA LOCAL is specified in a program, • in Natural for non-mainframe platforms, • when Natural Security is installed
OFF	All database field names in programming objects are considered long names, regardless of their length. This avoids possible misinterpretations of database field names in programs. You can use this setting to disallow the use of short names in general.

 Copyright © Software AG 2002

Background Information

When the Natural compiler resolves a database field (that is, a field defined in a DDM), the length of the field name is used to decide if the identifier represents a "db-short-name" or a "db-long-name". When the field name length is 2 characters a "db-short-name" reference is assumed, whereas all other identifiers are treated as "db-long-names".

According to the general Natural rules, you may not use "db-short-names" in your program, if a DEFINE DATA LOCAL was specified; neither to create the field list in the view definition nor in a search expressions of a FIND statement or to specify a read sequence control field in a READ or HISTOGRAM statement. All these restrictions are controlled by the Natural compiler, regardless if option DBSHORT is ON or OFF.

The purpose of DBSHORT is to change the compiler's behavior as follows:

if set to ON ,	everything works as described above.
if set to OFF,	every database field identifier is regarded as a "db-long-name", no matter of how many characters it consist. In other words, only the long name column in the DDM (captioned as "Name" in a DDM display) is considered to locate the referenced field and the DDM short names (captioned as "DB" in a DDM display) are completely disregarded.

The main reason for using DBSHORT=ON is when you have long names defined in a DDM with only 2 byte identifier length. At DDM generation, you may only create 2 byte long-names if the underlying database (you access with this DDM) is SQL (e.g. DB2). For all other database types, the attempt to define a long-field with 2 byte name length results in error SYSDDM4219 (SYSDDM utility).

However, when DBSHORT=OFF is set, the compiler does not check db-short-names in a DDM. This leads to syntax error NAT0981 if a db-short-field is used in a program.

PSIGNF - Internal Representation of Positive Sign of Packed Numbers

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

TSENABL - Applicability of TS Profile Parameter

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

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

The setting of the TSENABL option is currently not honored when a Natural object is executed. This will be corrected in the next major relase of Natural.

GFID - Generation of Global Format IDs

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

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

Rules for generating GLOBAL FORMAT-IDs in Natural Version 3.1.

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

For Natural nucleus internal system-file calls

`GFID=`*`abccddee`*

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

For user programs or Natural utilities

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

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

b) GFID=*axbbbbbc* for file number greater than 255 and Adabas Version lower than 6.2.

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

c) GFID=*abbbbbb* for Adabas Version 6.2 or higher.

 Copyright © Software AG 2002

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

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

LOWSRCE - Allow Lower-Case Source

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

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

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

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

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

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

  With LOWSRCE=ON, the compiler is **not** case sensitive and does not make a difference between lower/upper-case characters. This will lead to a syntax error because a duplicate definition of a variable is not allowed.
- Using the **session parameter** EM (Edit Mask) in an I/O statement or in a MOVE EDITED statement, there are characters which influence the layout of the data setting assigned to a variable (EM control characters), and characters which insert text fragments into the data setting. Example:

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

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

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

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

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

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

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

With LOWSRCE=OFF, the output is:

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

  ABC
```

With LOWSRCE=ON, the output is:

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

  ABC
```

FINDMUN - Detect Inconsistent Comparison Logic in FIND Statements

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

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

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

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

1.

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

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

2.

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

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

3.

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

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

 Copyright © Software AG 2002

4.

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

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

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

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

In Form 2:

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

into

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

In Form 3:

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

into

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

V22COMP - Compatibility Option for Inconsistent Version 2.2 Syntax - warning still valid???

This option will be available only for a limited period of time to allow a smooth transition. It will be removed again with a subsequent release of Natural.

The following inconsistent syntax constructions that were not intercepted by Version 2.2 lead to a syntax error with Version with Version 2.3 and above:

- DEFINE DATA: inconsistent number of decimal digits in constant setting,
- DEFINE DATA: redefinition of database array with variable index range,
- DEFINE WINDOW: specified window size smaller than minimum.

To allow you a smooth transition from Version 2.2, you can use this option.

ON	The above syntax constructions will **not** lead to a syntax error. Thus you are able to compile your existing programs with the current Natural version until you have adjusted them to the new syntax requirements.
OFF	The above syntax constructions will lead to a syntax error. This is the default.

Below, the above inconsistencies are explained in detail.

Decimal Digits of Constant Values

If the constant value specified after CONSTANT or INIT has more digits after the decimal point than the corresponding field, this does not lead to an error with Version 2.2. Now, such inconsistency leads to error NAT0094 at compilation.

Example:

```
DEFINE DATA LOCAL1 #FIELD (N2) INIT <12.25> /* no longer possibleEND-DEFINE
```

Redefinition of Database Arrays

To prevent referencing errors, it is no longer possible to specify a variable index range in the redefinition of a periodic-group field or multiple-value field.

Example:

```
DEFINE DATA LOCAL
1 MYVIEW VIEW OF EMPLOYEES
2 SALARY (I:I+2)
2 REDEFINE SALARY
3 MYSALARY (P9/I:I+2) /* no longer possible
END-DEFINE
```

The above redefinition has to be changed to:

```
3 MYSALARY (P9/1:3)
```

DEFINE WINDOW

The minimum possible size of a window is 2 lines by 10 columns without frame, and 4 lines by 12 columns with frame. The size of a window can be specified in the SIZE clause of the DEFINE WINDOW statement.

With Version 2.2, a window size smaller than the minimum could be specified: at runtime, the size of the window was then automatically set to the minimum possible number of lines and/or columns. Now, the specification of too small a window size leads to error NAT1167 at compilation.

Example:

```
DEFINE WINDOW XYZ
SIZE 4 * 8 FRAMED OFF      /* Version 2.2: Size set to 4 * 10 at runtime.
                           /* Version 2.3 and above: Syntax error.
```

Back to System Commands - Overview.

 Copyright © Software AG 2002

COMPOPT for UNIX/OpenVMS

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

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

option=value

When you issue the COMPOPT command without parameters, a screen is displayed on which you can set the options described below. Alternatively, you can set the options directly with the COMPOPT command using the keywords shown both in brackets below and on the COMPTOPT screen.

Example:

```
COMPOPT DBSHORT=ON
```

Note:
The default settings of the individual options are set with the respective profile parameters in the Natural parameter module.

Interpretation of Database Short Field Names (DBSHORT):

ON	Database field names in programming objects are considered long names (as defined in the corresponding DDM) - except 2-character field names, which are considered short names (as used by the underlying database system).
OFF	All database field names in programming objects are considered long names, regardless of their length. This avoids possible misinterpretations of database field names in programs.

Generation of Global Format IDs (GFID):

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

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

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

Allow Old Version 2.2 Syntax (V22COMP):

This option will be available only for a limited period of time to allow a smooth transition to Version 3.1 syntax and above. It will be removed again with a subsequent release of Natural.

The following inconsistent syntax construction not intercepted by Version 2.2 lead to a syntax error with Version 3.1 and above:

- DEFINE DATA: inconsistent number of decimal digits in constant setting.

To allow you a smooth transition from Version 2.2 to Version 3.1, you can use this option.

ON	The above syntax constructions will **not** lead to a syntax error. Thus you are able to compile your existing programs under Version 3.1 until you have adjusted them to the Version 3.1 requirements.
OFF	The above syntax constructions will lead to a syntax error.

For details on the above inconsistencies, please refer to your Natural Version 4.1.2 Release Notes. or mainframe description???

Back to System Commands - Overview.

 Copyright © Software AG 2002

COMPOPT for Windows

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

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

When you issue only the COMPOPT command itself, a screen is displayed where you can set the options described below.

Note:
The default settings of the individual options are set with the respective profile parameters in the Natural parameter module.
When you log on to another library, the COMPOPT options will be reset to their default settings.

Interpretation of Database Short Field Names - DBSHORT:

ON	Database field names in programming objects are considered long names (as defined in the corresponding DDM) - except 2-character field names, which are considered short names (as used by the underlying database system).
OFF	All database field names in programming objects are considered long names, regardless of their length. This avoids possible misinterpretations of database field names in programs.

Compatibility Option for Inconsistent Version 2.2 Syntax - V22COMP:

This option will be available for a limited period of time only to allow a smooth transition from Version 2.2. to Version 3.1 or above. It will be removed again with a subsequent release of Natural.

The following inconsistent syntax construction not intercepted by Version 2.2 leads to a syntax error with Version 3.1 or above.

- DEFINE DATA: inconsistent number of decimal digits in format definition and constant setting.

ON	The above syntax constructions will ***not*** lead to a syntax error. Thus you are able to compile your existing programs under version 3.1 or above until you have adjusted them to the version requirements.
OFF	The above syntax constructions will lead to a syntax error.

Generation of Global Format IDs - GFID:

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

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

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

option=value

Instead of changing an option on the screen, you can also specify it directly with the COMPOPT command. The keywords for the individual options are shown in parentheses (on the COMPOPT screen and in the above description).

Example:

COMPOPT DBSHORT=ON

Back to System Commands - Overview.

 Copyright © Software AG 2002

DCOMPARM

Note:
This command is for Mainframes only.

DCOMPARM [*server-ID*]

The system command DCOMPARM is used to display and modify the Natural parameters for a specified server ID in the system registry.

For more information, see DCOMPARM System Command in the NaturalX documentation.

Back to System Commands - Overview.

DEBUG

Note:
This command is for UNIX/OpenVMS and Windows platforms only.

```
DEBUG [object-name]
```

With the DEBUG command, you invoke the Natural debugging facility.

With the command, you specify the name of the object to be debugged.

The debugging facility is described in the Natural Debugging documentation.

Back to System Commands - Overview.

 Copyright © Software AG 2002

DELETE

Note:
This command is for Mainframes only, and this command is not available via the command line in a remote development environment.

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

The DELETE command is used to delete Natural programming objects from the Natural system file.

You can only delete programming objects which are stored in your current library.

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

This section covers the following topics:

- object-name
- object-type
- Selection List
- Safeguard Against Accidental Deletion
- Examples

object-name

As *object-name*, you specify the name(s) of the object(s) to be deleted.

In addition, you can specify whether an object's source code, its object module, or both are to be deleted:

SOURCE	source code
OBJECT	object module
BOTH	both, source code and object module

If you make no specification, BOTH applies.

A SOURCE/OBJECT/BOTH specification applies for all subsequent object names that is, until the next SOURCE/OBJECT/BOTH specification.

To delete all objects whose names begin with a specific string of characters, you can use asterisk notation for the *object-name*.

object-type

In conjunction with asterisk notation for the *object-name*, you can also specify an *object-type* if you wish to delete only objects of a specific type.

The possible settings for *object-type* are the same as for the system command EDIT. In addition, you can specify the settings **X** (= global, local and parameter data areas) and **U** (= subprograms, subroutines and helproutines).

Note:
If you specify the full names of individual objects, you need not specify their object types.

Selection List

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

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

Safeguard Against Accidental Deletion

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

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

 Copyright © Software AG 2002

Examples

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

```
DELETE TOM DICK HARRY
```

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

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

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

```
DELETE
```

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

```
DELETE TYPE M SOURCE *
```

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

```
DELETE TYPE GLA D*
```

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

```
DELETE OBJECT YYZ*
```

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

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

If TOM, DICK and HARRY are not maps, or if JOHN and PAUL are not programs, they will not be deleted.

Back to System Commands - Overview.

DUMP

Note:
This command is for Mainframes only.

```
DUMP  [ZAPS]
```

The DUMP command is used to provide information for Software AG technical support personnel in order to locate an error that caused an abnormal termination (abend) of the Natural system.

DUMP displays abend information (core contents).

DUMP ZAPS displays a list of all ZAPs applied.

Forward this information to Software AG technical support for error diagnosis and correction.

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

Back to System Commands - Overview.

 Copyright © Software AG 2002

EDIT

- For Mainframes
- For UNIX/OpenVMS
- For Windows

EDIT for Mainframes

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

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

Below is information on:

- object-type
- object-name
- library-id

object-type

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

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

- If the object to be edited is a LOCAL data area, GLOBAL data area or PARAMETER data area, the data area editor will be invoked.
- If the object to be edited is a MAP, the map editor will be invoked.
- All other object types - CLASS | 4, PROGRAM, SUBPROGRAM, SUBROUTINE, HELPROUTINE, COPYCODE, TEXT, DESCRIPTION - are edited in the program editor. (A DESCRIPTION is a program description as stored and maintained in the Predict Data Dictionary; an object of this type can only be edited if Predict is installed.)

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

object-name

With the EDIT command, you specify the name of the object you wish to edit (maximum 8 characters); Natural will then load the object into the edit work area of the appropriate editor and set the object name for a subsequent SAVE, CATALOG, STOW command.

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

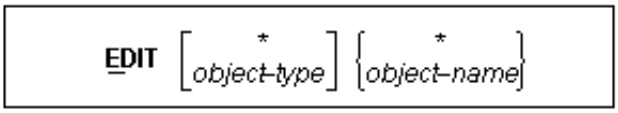
For EDIT DESCRIPTION, the *object-name* must be the name as defined as a Natural member in the Predict program definition.

library-id

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

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

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

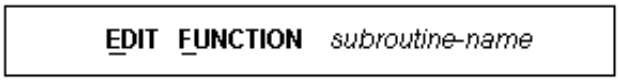


If you do not remember the name of the object you wish to edit, you can use this form of the EDIT command to display a list of objects, and then select from the list the desired object.

EDIT * displays a list of all objects in your current library.

EDIT *object-type* ***** displays a list of all objects of that type in your current library.

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



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

Example:

```
DEFINE SUBROUTINE ABC
  ...
  END-SUBROUTINE
  END
```

Assuming that the above subroutine has been saved under the object name "SUB", you may edit subroutine ABC either by issuing the command EDIT S SUB or by EDIT F ABC.

 Copyright © Software AG 2002

Back to System Commands - Overview.

EDIT for UNIX/OpenVMS

Syntax 1

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

The EDIT command is used to invoke a Natural editor for the purpose of editing a Natural object.

object-type

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

The EDIT VIEW only works in the current library and when an *object-name* is specified.

If you are in the program editor and enter EDIT object-name (whereas the Natural object is of type copycode, program, subprogram,subroutine, class, text, helproutine), a parallel edit session will be opened. See also the Program Editor, editor command NEXT.

object-name

As *object-name* you specify the name of the object to be edited (maximum 8 characters; exception: view names contain maximum 32 characters); Natural will then load the object into the edit work area of the appropriate editor and set the object name for a subsequent SAVE, CATALOG, or STOW command.

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

library-id

As *library-id*, you specify the library in which the object to be edited is contained. The *library-id* need only be specified if the object to be edited is contained in a library other than the one you are currently logged on to.

Under Natural Security, you cannot specify a *library-id*; that is, you can edit only objects which are stored in your current library.

 Copyright © Software AG 2002

Syntax 2

EDIT FUNCTION *subroutine-name*

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

Example:

```
DEFINE SUBROUTINE ABC
...
END-SUBROUTINE
END
```

Assuming that the above subroutine has been saved under the object name "SUB", you may edit subroutine ABC either by issuing the command EDIT S SUB or by EDIT F ABC.

Back to System Commands - Overview.

EDIT for Windows

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

The EDIT command is used to invoke a Natural editor for the purpose of editing a Natural object.

object-type

```
{ CLASS|4
  COPYCODE
  DIALOG|3
  GLOBAL
  HELPROUTINE
  LOCAL
  MAP
  PARAMETER
  PROGRAM
  SUBPROGRAM|N
  SUBROUTINE
  TEXT
  FUNCTION
  VIEW }
```

As *object-type* you specify the type of object to be edited.

object-name

As *object-name* you specify the name of the object to be edited (maximum of 8 characters); Natural will then load the object into the edit work area of the appropriate editor and set the object name for a subsequent SAVE, CATALOG, or STOW command.

If you enter the EDIT command without any parameters, the objects marked in the "Objects" or "DDMs" window are loaded into the appropriate editors.

library-id

As *library-id*, you specify the library in which the object to be edited is contained. The *library-id* need only be specified if the object to be edited is contained in a library other than the one you are currently logged on to.

Under Natural Security, you cannot specify a *library-id*; that is, you can edit only objects that are stored in your current library.

 Copyright © Software AG 2002

Edit Function

EDIT FUNCTION *subroutine-name*

The EDIT FUNCTION command can be used to edit a subroutine using the subroutine name (not the object name).

Example:

```
DEFINE SUBROUTINE ABC
...
END-SUBROUTINE
END
```

Assuming that the above subroutine has been saved under the object name "SUB", you can edit subroutine ABC by issuing either the command EDIT S SUB or the command EDIT F ABC.

Back to System Commands - Overview.

EDT

Note:
This command is for Mainframes only. This command is not available via the command line in a remote development environment.

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

It is recommended that the EDIT command be used instead of the EDT command.

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

Use ".E" to terminate EDT mode.

This section covers the following topics:

- Syntax Explanation
- EDT Subcommands
- EDT Function Keys

Syntax Explanation

object-name

As *object-name*, you enter the name of the object to be edited (maximum 8 characters). If *object-name* is entered, Natural will load the object into the source work area and set the object name for a subsequent SAVE, CATALOG, or STOW command.
If you enter the EDT command without an object name and there is no object in the source work area, you will be prompted with line 0010 to enter an object.
If you do not specify an object name and there is an object in the source program work area, the first lines of that object will be displayed.

library-id

If the object to be edited is contained in a library other than the one to which you are currently logged on, you have to specify the ID of the library in which the object is contained.

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

Entering a library ID is not permitted if Natural Security is active.

 Copyright © Software AG 2002

EDT Subcommands

The following subcommands may be used during line editing:

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

Copyright © Software AG 2002

EDT Function Keys

The following PF keys can be used during line editing:

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

Back to System Commands - Overview.

 Copyright © Software AG 2002

EXECUTE

- For Mainframes
- For UNIX/OpenVMS and Windows

EXECUTE for Mainframes

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

The EXECUTE command is used to execute a Natural object module.

The object module must have been cataloged (that is, stored in object form) in the Natural system file or linked to the Natural nucleus.

The keyword EXECUTE is optional; it is sufficient to specify the name of the program to be executed.

The execution of an object module does not affect the source program currently in the editor work area.

When entered in the command line of the program editor, the system command EXECUTE **must not** be abbreviated to EX, as the program editor would interpret this as the editor command EX.

This section covers the following topics:

- Syntax Explanation
- Passing Parameters to the Program

Syntax Explanation

REPEAT

If the program being executed produces multiple screen output and you wish the screens to be output one after another without intervening prompts, you specify the keyword REPEAT together with the keyword EXECUTE.

program-name

The name of the program to be executed. If you do not specify a *library-id*, Natural can only execute the specified program if it is stored either in your current library or in the current steplib library (the default steplib is "SYSTEM").

library-id

If the program is stored in another library, specify the *library-id* of that library. In this case, the program can only be executed if it actually stored in the specified library.

A *library-id* that begins with "SYS" must not be specified (except "SYSTEM").

Under Natural Security, a library ID cannot be specified.

Passing Parameters to the Program

When you execute a program by specifying the program name without the keyword EXECUTE, you may pass parameters to the program. These parameters will be read by the first INPUT statement in the executed program.

You can specify the parameters as positional parameters or as keyword parameters, with the individual specifications separated from one another by blanks or the input delimiter character (as specified with the **session parameter** ID).

Note:
The parameter settings are always converted to upper case (regardless of the terminal command %L or the **profile parameter** LC=ON).

Examples of EXECUTE Command:

```
EXECUTE PROG1

EXECUTE PROG1 ULIB1

PROG1

PROG1 VALUE1 VALUE2 VALUE3

PROG1 VALUE1, VALUE2, VALUE3

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

PROG1 PARM3=VALUE3 PARM1=VALUE1 VALUE2
```

Back to System Commands - Overview.

 Copyright © Software AG 2002

EXECUTE for UNIX/OpenVMS and Windows

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

The EXECUTE command is used to execute a Natural object module.

The object module must have been cataloged (that is, stored in object form) in a Natural library or linked to the Natural nucleus.

The keyword EXECUTE is optional; it is sufficient to specify the name of the program to be executed. The execution of an object module does not affect the source program in the currently active window.

If the program being executed produces multiple screen output and you wish the screens to be output one after another without intervening prompts, specify the keyword REPEAT together with the keyword EXECUTE.

program-name

The name of the program to be executed. If you do not specify a *library-id*, Natural can only execute the specified program if it is stored either in your current library or in the current steplib library (the default steplib is "SYSTEM").

library-id

If the program is stored in another library, specify the *library-id* of that library. In this case, Natural can execute the program only if it is actually stored in the specified library.

Under Natural Security, you cannot specify a *library-id*; that is, you can execute only programs that are stored in your current library.

Passing Parameters to the Program

When a Natural program is executed by specifying the program name only, input parameters may be passed to the Natural program. These parameters will be read by the first INPUT statement in the executed Natural program. The parameters can be specified as either positional parameters or keyword parameters with each setting separated by the input delimiter character.
If one of the parameters passed contains blanks, the transfer will only be executed if directly after the program name an input delimiter is set.

If one of the parameters is a string which contains blankes, the tranfer will only be executed if directly after the program name an input delimiter is set.

Examples of EXECUTE Command:

EXECUTE PROG1

EXECUTE PROG1 ULIB1

PROG1

PROG1 VALUE1 VALUE2 VALUE3

PROG1 VALUE1, VALUE2, VALUE3

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

PROG1 PARM3=VALUE3 PARM1=VALUE1 VALUE2

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

Back to System Commands - Overview.

 Copyright © Software AG 2002

FIN

Note:
This command applies to all platforms.

```
FIN
```

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

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

Back to System Commands - Overview.

GLOBALS

Note:
This command applies to all platforms. If you are using remote development facilities, the GLOBALS command only affects the current environment.

```
GLOBALS [parameter...]
```

The GLOBALS command is used to set Natural session parameters.

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

Parameter settings can be supplied in any order and must be separated by a blank. If more parameters are specified than will fit on one command line, several GLOBALS commands must be issued.

If the GLOBALS command is entered without parameters, a dialog box is displayed from which you can select parameter groupings and modify the settings.

The individual parameters are described in the section Session Parameters of the Natural Reference documentation.

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

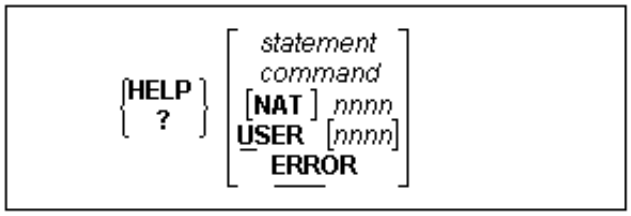
Back to System Commands - Overview.

 Copyright © Software AG 2002

HELP

- For Mainframes
- For UNIX/OpenVMS and Windows

HELP for Mainframes



The HELP command invokes the Natural help facility. This command retrieves information concerning the current development environment.

The following options are available with the HELP command:

Option	Function
HELP	Displays the help menu.
HELP *statement*	Displays help information for the specified *statement*.
HELP *command*	Diplays help information for the specified *command*.
HELP [NAT]*nnnn*	Entering HELP and a number (up to 4 digits, optionally prefixed by "NAT") displays the detailed message text for the Natural error condition associated with that number, that is, the long text of the Natural *system* error message NAT*nnnn.*
HELP USER*nnnn*	Displays the long text of the *library-specific* error message number *nnnn* in the current library.
HELP USER	Displays a selection list of **all** library-specific messages in the current library.
HELP ERROR	Displays the long text for the error that occurred last.

Back to System Commands - Overview.

HELP for UNIX/OpenVMS and Windows

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

The HELP command invokes online help for error messages.

Note for users of remote development facilities: This command retrieves information concerning the current development environment.

The following options are available with the HELP command:

Option	Function
HELP	Displays the help menu.
HELP [NAT]*nnnn*	If you enter HELP and a number (up to 4 digits, optionally prefixed by "NAT"), the detailed message text for the Natural error condition associated with that number will be displayed, i.e., the long text of the Natural *system* error message NAT*nnnn*.
HELP USER*nnnn*	Displays the long text of the *library-specific* error message number *nnnn* in the current library.

Back to System Commands - Overview.

 Copyright © Software AG 2002

INPL

- For Mainframes
- For UNIX/OpenVMS
- For Windows

INPL for Mainframes

```
INPL [R]
```

If you enter the INPL command without any parameters, the INPL utility will be invoked. This utility is **only** used for the loading of Software AG installation datasets into the system files as described in the Natural INPL Utility documentation.

Apart from that, you use the utility NATLOAD to load objects into the system files.

The INPL functions used for the installation are described in the INPL online help. Apart from that, the INPL command provides the following special function:

INPL R - Natural Security Recover

This option is only available if Natural Security is installed.

"INPL R" is used to reset the access to the Natural Security library SYSSEC to its original status.

If you issue "INPL R", the user DBA, the library SYSSEC, and the link between the two will be redefined as after the initial installation, while all other links to SYSSEC will be cancelled. See also Inaccessible Security Profiles in the section Countersignatures of the Natural Security documentation.

Back to System Commands - Overview.

INPL for UNIX/OpenVMS

```
INPL [R]
```

If you enter the INPL command without any parameters, the INPL utility will be invoked. This utility is *only* used for the loading of Software AG installation datasets into the file system as part of the installation process (as described in the Setup and Installation of Natural under UNIX or Installing and Setting Up Natural on OpenVMS Documentation).

Apart from that, you use the utility NATLOAD in library SYSUNLD to load objects into the file system.

In addition, the INPL command provides the following special function:

INPL R - Natural Security Recover

This option is only available if Natural Security is installed.

"INPL R" is used to reset the access to the Natural Security library "SYSSEC" to its original status.

If you issue "INPL R", the user "DBA", the library "SYSSEC", and the link between the two will be redefined as after the initial installation, while all other links to "SYSSEC" will be cancelled. See also Inaccessible Security Profiles in the section Countersignatures of the Natural Security documentation.

Back to System Commands - Overview.

 Copyright © Software AG 2002

INPL for Windows

```
INPL [R]
```

If you enter the INPL command without any parameters, the INPL utility will be invoked. This utility is *only* used for the loading of Software AG installation datasets into the file system as part of the installation process (as described in your Natural Installation Guide and the INPL Utility documentation).

Apart from that, you use the utility SYSPAUL (which is described in the Natural Operations documentation) to load objects into the file system.

In addition, the INPL command provides the following special function:

INPL R - Natural Security Recover

Note:
This option is only available if Natural Security is installed.

"INPL R" is used to reset the access to the Natural Security library "SYSSEC" to its original status.

If you issue "INPL R", the user "DBA", the library "SYSSEC", and the link between the two will be redefined as after the initial installation, while all other links to "SYSSEC" will be cancelled. See also Inaccessible Security Profiles in the section Countersignatures of the Natural Security documentation.

Back to System Commands - Overview.

KEY

- For Mainframes
- For UNIX/OpenVMS

KEY for Mainframes

```
KEY [ ON
      OFF
      { { PA n } = { ON      } }
      { { PF n }   { OFF     } }
      { { CLR  }   { command } } ... ]
```

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

This is possible for the following keys: the PA keys PA1 to PA3, the PF keys PF1 to PF24, and the CLEAR key.

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

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

Natural will executed the assigned command whenever you press the corresponding key in command mode (NEXT prompt).

Note:
Assignments made with the system command KEY are totally independent of assignments made with a SET KEY statement in a program.
Function-key assignments can also be made by the Natural administrator via the **profile parameter** KEY.

The section below covers the following topics:

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

Natural Utilities for Mainframes
Assigning Commands

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

 Copyright © Software AG 2002

To assign a different command to a key, you overwrite the existing entry in the "Command" column.

To delete an command assignment, you delete the entry in the "Command" column or overwrite it with blanks.

You can also assign commands to individual keys by specifying them directly with the KEY command; that is, you specify KEY *key=command* (*key* being the name of the key and *command* the name of the command that is to be assigned to the key). For example:

```
KEY PF1=CLEAR PF2=MENU
```

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

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

Activating/Deactivating All Keys - KEY ON/OFF

With the command KEY OFF you deactivate all function-key assignments: if you the press a function key, Natural will return an appropriate message indicating that the key is not active.

With the command KEY ON you re-activate all function-key assignments that have previously been deactivated with KEY OFF.

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

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

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

With the command KEY *key*=OFF you deactivate the command assigned to a specific *key*. For example:

```
KEY PF24=OFF
```

With the command KEY *key*=ON you re-activate a previously deactivated command assignment. For example:

```
KEY PF24=ON
```

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

Back to System Commands - Overview.

KEY for UNIX/OpenVMS

```
KEY [ ON
      OFF
      { { PAn } = { ON      } }
      { { PFn }   { OFF     } }
      { { CLR }   { command } } ... ]
```

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

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

This is possible for the following keys: the PA keys PA1 to PA3, the PF keys PF1 to PF24, and the CLEAR key.

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

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

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

Note:
Assignments made with the system command KEY are totally independent of assignments made with a SET KEY statement in a program.

Function-key assignments can also be made by the Natural administrator via the **profile parameter** KEY.

Assigning Commands

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

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

To delete an command assignment, you delete the entry or overwrite it with blanks.

You can also assign commands to individual keys by specifying them directly with the KEY command; that is, you specify **KEY***key***=***command* (*key* being the name of the key and *command* the name of the command that is to be assigned to the key). For example:

KEY PF1=CLEAR

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

 Copyright © Software AG 2002

KEY PF13='UPDATE OFF'

Activating/Deactivating All Keys - KEY ON/OFF

With the command **KEY OFF** you deactivate all function-key assignments: if you press a function key, Natural will return an appropriate message indicating that the key is not active.

With the command **KEY ON** you re-activate all function-key assignments that have previously been deactivated with KEY OFF.

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

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

With the command **KEY***key***=OFF** you deactivate the command assigned to a specific *key*. For example:

KEY PF24=OFF

With the command **KEY***key***=ON** you re-activate a previously deactivated command assignment. For example:

KEY PF24=ON

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

Back to System Commands - Overview.

LAST

Note:
This command applies to all platforms.

```
LAST [ -  ]
     [ -nn ]
     [ *   ]
     [ ?   ]
```

With the LAST command, you can have the system command(s) that was/were last executed displayed. Moreover, you can have the displayed command(s) executed again. You can also overwrite it before they are executed.

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

With the following options of the LAST command, you can have one of the last commands written into the command line or NEXT line:

LAST	**For Mainframes:** The command that was issued last will be placed in the command line or NEXT line. **For UNIX/OpenVMS:** The system command that was issued last is placed in the Direct Command window and can be executed. **For Windows:** The system command that was issued last is placed in a dialog box and can be executed.
LAST -	**For Mainframes:** The command that was issued last will be placed in the command line or NEXT line. If you enter "LAST -" again, the last but one command will be placed in the command line or NEXT line. By repeatedly entering "LAST -", you can thus "page" backwards command by command. (Instead of repeatedly entering it by hand, you can assign "LAST -" to a PF key via the system command KEY.) **For UNIX/OpenVMS:** The system command that was issued last is placed in the Direct Command window and can be executed. If you enter "LAST -" again, the last but one command will be placed in the Direct Command window. By repeatedly entering "LAST -", you can thus "page" backwards command by command. (Instead of repeatedly entering it by hand, you can assign "LAST -" to a PF key via the system command KEY.) **For Windows:** The system command that was issued last is placed in a dialog box and can be executed. If you enter "LAST -" again, the last but one command will be placed in a dialog box. By repeatedly entering "LAST -", you can "page" backwards command by command.
LAST -*nn*	Natural "remembers" up to the last 20 commands that were issued; *nn* must therefore not be greater than 20. **For Mainframes:** The last command but *nn* will be placed in the command line or NEXT line. **For UNIX/OpenVMS:** The nnth previous system command is written into the Direct Command window and can be executed. **For Windows:** The nnth previous system command is written into a dialog box and can be executed.

 Copyright © Software AG 2002

LAST *	**For Mainframes:** If you enter "LAST *", a window will be displayed showing the last 9 system commands that were issued. In the window, you can select the command(s) you wish to be executed again: ● To execute a **single** command again, either mark the command with a cursor and press PF5, or mark the command with any character and press ENTER. ● To execute **several** commands again, mark them with numbers in the order in which you wish them to be executed and press ENTER, the commands will then be executed in ascending order of numbers. **Note**: Before you have a command executed again, you can overwrite it. If a command is too long to be fully visible in the window, press PF11 for a larger window; to switch back to the smaller window, press PF10. **For UNIX/OpenVMS:** If you enter "LAST *", a window will be displayed showing the last 20 commands that were issued. Use PF8 and PF7 to scroll forward and backward if more than 10 commands are displayed: ● To execute a **single** command again, either mark the command with the cursor and press F5, or mark the command with any character and press ENTER. ● To execute **several** commands again, mark them with numbers in the order in which you wish them to be executed and press ENTER, the commands will then be executed in ascending order of numbers. **For Windows:** When you enter "LAST *", a dialog box is displayed showing the last 20 system commands that were issued. ● To execute the commands again, copy the **requested** commands into the list box ’Selected Commands’. ● The selected commands in the list box can be modified before executing them.
LAST ?	**For Mainframes and UNIX/OpenVMS only:** With "LAST ?" you can call the Help function for the LAST command.

Back to System Commands - Overview.

LASTMSG

Note:
This command applies to all platforms.

```
LASTMSG
```

With the LASTMSG command, you can display additional information about the error situation which has occurred last.

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

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

Mark one of these messages with the cursor (for Mainframe and UNIX/OpenVMS) or with a double-click or select one and press the ’Details Button’ (for Windows) to display the following information on the corresponding error:

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

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

Note for Natural RPC:
In the case of an error on the server, the following error information is not displayed: database ID, file number, date and time.

Note for Windows platforms:
This command is also available in a remote session.
All information can be read in batch mode.

Back to System Commands - Overview.

 Copyright © Software AG 2002

LIST

- For Mainframes
- For UNIX/OpenVMS
- For Windows

LIST for Mainframes

The LIST command is used to list one or more objects which are contained in the current library.

The numerous options of the LIST command are explained below.

Since LIST can display long lines containing up to 244 characters, set the profile parameter LS as big as possible, if possible LS=250.

This section covers the following topics:

- Syntax Diagrams
- Functions
- Options
- List of Objects
- List of Source
- Subcommands in Expand Object

Syntax Diagrams

```
                 [ SEQUENTIAL ] [object-type] object-name [options]
                 DIRECTORY [object-name]
LIST             EXTENDED [extended-type] object-name
                 DDM [ddm-name]
                 XREF
```

object-type

```
*
PROGRAM
MAP
DATA-AREAS
   GLOBAL
   LOCAL
   PARAMETER
ROUTINES
   HELPROUTINE
   N SUBPROGRAM
   SUBROUTINE
COPYCODE
TEXT
RECORDING
DIALOG
CP PROCESSOR
4 | CLASS
```

extended-type

```
*
SUBROUTINE
4 | CLASS
```

 Copyright © Software AG 2002

options

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

formatted-option

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

expand-option

```
EXPAND [FORMATTED ] [ COMMENTS ] [expand-type...10] object-name
                    [    n     ]
```

Functions

This section covers the following topics:

- Displaying an Individual Source
- Displaying Directory Information
- Displaying Long Names of Cataloged Subroutines and Classes
- LIST SEQUENTIAL
- Displaying DDMs
- Displaying XREF Data
- LIST EXT extended-type object-name and LIST object-type object-name

Displaying an Individual Source

LIST	If you enter only the LIST command itself, without any parameters, the contents of the source work area will be listed.
LIST *object-name*	If you enter a single *object-name* with the LIST command, you need not specify the *object-type*; the object's source code will be listed.

Displaying Directory Information

LIST DIR	This command displays the directory information on the object currently in the source work area.
LIST DIR *object-name*	This command displays the directory information on the specified object. To display the directory information of several objects, you use asterisk notation for the *object-name*.
LIST *object-name* WITH DIR	This command first displays the directory information on the specified object and then lists the source code of the object.

Displaying Long Names of Cataloged Subroutines and Classes

LIST EXT [extended-type] *object-name*	This command displays a list of the long names of cataloged subroutines and classes and directory information. The same name options as for LIST *object-name* apply.

 Copyright © Software AG 2002

LIST SEQUENTIAL

LIST SEQUENTIAL	Normally, when you use asterisk or wildcard notation for the *object-name*, you get a list of all objects that meet the specified selection criteria. On the list you can then select objects for display by marking them with the function code "LI" (see below). If you specify LIST SEQ(UENTIAL), the selection list will be suppressed, and the sources of all objects that meet the selection criteria will be displayed sequentially, i.e. one after the other.

Displaying DDMs

LIST DDM	This command displays a list of all DDMs.
LIST DDM *ddm-name*	If you specify a single DDM name, the specified DDM will be displayed. For the *ddm-name* you can use the same range notations (*, ? , <, >) as for *object-name* to display a list of a certain range of DDMs.

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

Displaying XREF Data

LIST XREF	This command displays all active cross-reference data for the current library. This command is only available if Predict with active cross-references is installed. See the Predict documentation.

LIST EXT extended-type object-name and LIST object-type object-name

If you specify an *object-type*, you must also specify an *object-name*.

To have all objects in the current library listed, you specify "*" for the *object-type*, but no *object-name*.

To have all objects of a certain type listed, you specify a certain *object-type* and "*" for the *object-name*.

If you wish a certain range of objects to be listed, you can use asterisk notation and wildcard notation for the *object-name*:

- Asterisk notation is the option to specify an asterisk (*) in the *object-name*: the asterisk stands for any string of characters of any length.
- Wildcard notation is the option to specify a question mark (?) in the *object-name*: the question mark stands for any single character.

One or more asterisk and wildcard notations can be combined in an *object-name*.

For a list of all objects from a specific start setting or until a specific end setting, you can use the notation ">" or "<" respectively.

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

Examples:

LIST *	lists all objects in the current library - regardless of their types.
LIST S *	lists all subroutines in the current library.
LIST SYS*	lists all objects (of any type) whose names begin with "SYS".
LIST M SYS*	lists all maps whose names begin with "SYS".
LIST C *CODE	lists all copycodes whose names end with "CODE".
LIST NAT*AL	lists all objects whose names begin with "NAT" and end with "AL" no matter which and how many other characters are between "NAT" and "AL" (this would include the names "Natural" and "NATIONAL" as well as "NATAL").
LIST DOO?	lists all objects with 4-character names beginning with "DOO" (this would include the names "DOOR" and "DOOM", but not "DOO" or "DOODLE").
LIST M NAT?AL	lists all maps whose names begin with "NAT" and end with "AL" with exactly one character are between "NAT" and "AL" (this would include the names "NAT1AL" and "NAT2AL", but not "NATAL" or "NATIONAL").
LIST M *1*	lists all maps whose names contains a "1".
LIST M F>	lists all maps, starting from the first one whose name begins with "F".
LIST M MA<	lists all maps, from the first one until the one named "MA" (if present).
LIST EXT *	lists the long names of all subroutines/classes in the current library - regardless of their type.

 Copyright © Software AG 2002

Options

- NUMBERS OFF
- FORMATTED
- SETTINGS
- FIELDS
- EXTFIELDS
- RULES/INLINERULES/FREERULES/AUTORULES

NUMBERS OFF

By default, the source code of an object will be listed with source-code line numbers. To list it without line numbers, specify the NUMBERS OFF option. (See also subcommands NUMBERS ON/NUMBERS OFF in the section Subcommands for Listed Source.)

FORMATTED

This option applies only to data areas and maps:

- If you specify this option for a data area, the data area will be displayed formatted; that is, the display resembles a DEFINE DATA statement. (See also subcommand FORMAT in the section List of Source.) By default, data areas are displayed unformatted; that is, the display resembles that in the data area editor.
- If you specify this option for a map, the map *layout* will be displayed, that is, the map as it is displayed to the users at runtime. In addition, you may specify filler characters *c* for input fields (AD=A and AD=M) and output fields (AD=O) to make these fields visible. You may specify any character as filler character. (See also the subcommands LAYOUT and FORMAT in the section List of Source.)

SETTINGS

This option applies only to maps; it causes the map settings of the map to be displayed.

FIELDS

This option only applies to maps; it causes the field summary, that is, the list of fields in the map, to be displayed.

EXTFIELDS

This option only applies to maps; it causes the extended field editing information for all map fields to be displayed.

RULES/INLINERULES/FREERULES/AUTORULES

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

RULES displays all processing rules. INLINERULES displays only the inline rules. FREERULES displays only the free rules. AUTORULES displays only the automatic rules.

Copyright © Software AG 2002

List of Objects

- Scrolling the Selection List of Objects
- New Criteria for the Selection List
- Information Displayed on the Selection List
- Items Intensified on the Selection List
- Subcommands for a List of Objects
- Selecting an Object from the Selection List

Scrolling the Selection List of Objects

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

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

New Criteria for the Selection List

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

Information Displayed on the Selection List

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

Press PF11 to display more information on source and cataloged objects.

Items Intensified on the Selection List

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

 Copyright © Software AG 2002

Subcommands for a List of Objects

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

Code	Function
SC	List only objects containing a scan setting (cannot be used if SHORT list is active).
SC OFF	Switch off scan mode.
SHORT	Display a short list of objects, i.e., display only the object names (cannot be used if SC mode is active).
LONG	Switch to "normal" list.
PRINT	Print the list of objects.
EXTENDED	Display the list of long names of subroutines/classes.
ALL fx	Enter the function code "fx" (where "fx" is a valid function code for a listed object) for all displayed objects.
+	Scroll one page forward.
-	Scroll one page backward.
++	Scroll to the end (bottom) of the object list.
--	Scroll to the beginning (top) of the object list.

Copyright © Software AG 2002

Selecting an Object from the Selection List

To select an object from the selection list for a function, you simply mark the object with the appropriate function code in the left-hand column (titled "Cmd").

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

Code	Function
?	A window will be displayed which shows all the functions available for the marked object. The window will only list those functions that are actually available for the selected object (for example, if the object is a subroutine, it cannot be run; if the object is only available in source form, it cannot be executed). From the window you can select the function to be performed on the marked object.
CA	Compile the object and store it in object form (equivalent to the system command CATALOG).
DE	Delete the object (equivalent to the system command DELETE).
DL	Download object from mainframe to personal computer (only available if Natural Connection is installed).
ED	Edit the object's source (equivalent to the system command EDIT).
EX	Execute the object (equivalent to the system command EXECUTE).
LD	List directory information (equivalent to LIST DIR *object-name*).
LE	List in expanded form (equivalent to LIST *object-name* EXPAND *).
LF	Display a data area or map formatted (equivalent to LIST *object-name* FORMATTED).
LI	List the object's source.
RE	Rename the object (equivalent to RENAME).
PR	Print the object's source.
RU	Run (that is, compile and execute) the object's source (equivalent to the system command RUN).
ST	Stow the object in source and object form (equivalent to the system command STOW).
UC	Delete the object module (uncatalog).
.	Exit.

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

 Copyright © Software AG 2002

List of Source

- Subcommands for Listed Source
- FORMAT
- Cursor-Sensitive Object Selection
- expand-option

Subcommands for Listed Source

When you have the source code of an object listed, you can enter in the command line one of the commands described below.

Subcommand	Function
+	Scrolls one page forward.
-	Scrolls one page backward.
++	Scrolls to the end (bottom) of the source.
BOTTOM	
--	Scrolls to the beginning (top) of the source.
TOP	
+*n*	Scrolls *n* lines forward.
-*n*	Scrolls *n* lines backward.
nnnn	Scrolls to line number *nnnn*.
EXPAND	See Expand-Option.
FIELDS	Applies to maps only: displays the field summary; that is, the list of fields in the map.
FIND [ABSOLUTE] *setting*	Displays only those source lines which contain the specified *setting*. If you enter only the command FIND itself, a window will be displayed in which you can enter the *setting* to be sought for and specify whether the search is to be absolute or not. By default, the search will not be absolute; that is, the *setting* will only be found if it is an isolated word. If you specify "ABS(OLUTE)" after the command, the search will be absolute; that is, the *setting* will also be found if it is part of a larger string of characters.
FORMAT	Applies to data areas and maps only: displays "formatted" data area or map, and items related to the map.
LAYOUT	Applies to maps only: displays the map layout; that is, the map will be displayed as it is displayed to the users at runtime.
NUMBERS ON	Displays the source **with** source-code line numbers.
NUMBERS OFF	Displays the source **without** source-code line numbers.
PRINT	Prints the listed source.

Subcommand	Function
REF [ABSOLUTE] *setting*	Displays the line numbers of the source-code lines which contain the specified *setting*. If you enter only the command REF itself, a window will be displayed in which you can enter the *setting* to be sought for and specify whether the search is to be absolute or not. By default, the search will not be absolute; that is, the *setting* will only be found if it is an isolated word. If you specify "ABS(OLUTE)" after the command, the search will be absolute; that is, the *setting* will also be found if it is part of a larger string of characters.
RULES	Applies to maps only: displays the processing rules used by the map (the rules are displayed in order of fields to which they are assigned, and per field in order of rank).
SCAN [ABSOLUTE] *setting*	Displays all lines intensified which contain the specified *setting*. The source will be scrolled to the first line that contains the *setting*. If you enter only the command SCAN itself, a window will be displayed in which you can enter the *setting* to be sought for and specify whether the search is to be absolute or not. By default, the search will not be absolute; that is, the *setting* will only be found if it is an isolated word. If you specify "ABS(OLUTE)" after the command, the search will be absolute; that is, the *setting* will also be found if it is part of a larger string of characters.
SCAN= or SC=	Executes the last SCAN command again.
SETTINGS	Applies to maps only: displays the map settings of the map.
ZOOM [*expand-type*...10] *object-name*	Specifying a single *object-name* with the ZOOM command has the same effect as marking the name in the listed source with the cursor (see the section Cursor-Sensitive Object Selection): the selected object will be displayed in a window. If you use asterisk/wildcard notation for the *object-name*, all selected objects will be displayed in a window in the sequence in which they are referenced in the listed source; see also below. The specification of an *expand-type* is the same as for the EXPAND option. For an object displayed within a window invoked by ZOOM, the same subcommands (except PRINT, EXPAND and ZOOM) are available as for the normal listed source. Moreover, if you have used asterisk or wildcard notation and several objects are displayed, you can use the commands "NEXT" and "PREV" (or PF4 and PF5) to move from one object in the window to the next one or previous one respectively.
.	Exit.

 Copyright © Software AG 2002

FORMAT

This subcommand only applies to data areas and maps.

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

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

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

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

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

Additional Subcommands for Map Layout

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

Copyright © Software AG 2002

Additional Subcommands for Field Summary List

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

Additional Subcommands for Processing Rules

SCAN [ABSOLUTE] *setting*	Same as for listed source.
SCAN =	Same as for listed source.

Cursor-Sensitive Object Selection

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

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

 Copyright © Software AG 2002

expand-option

```
EXPAND [FORMATTED] [COMMENTS n] [expand-type...10] object-name
```

With the EXPAND option, you can have the sources of other objects referenced by the listed object - copycodes, data areas, maps, helproutines, external subroutines, subprograms, FETCHed programs, error messages - listed **within** the source of the listed object. This option is particularly useful in batch mode.

For example, if a listed source program contains an INCLUDE statement, you can have the source code of the included copycode listed within the listed source program immediately after the INCLUDE statement.

Objects listed within a source will be referred to as *expand objects* in the explanations below.

EXPAND FORMATTED

The FORMATTED option is only relevant for data areas and maps listed within a source.

For data areas, the following applies:

- If FORMATTED is not specified, the display of the data area will resemble that in the data area editor
- If FORMATTED is specified, the display of the data area will resemble a DEFINE DATA statement.

For maps, the following applies:

- If FORMATTED is not specified, the map *source* will be listed.
- If FORMATTED is specified, the map *layout* will be displayed (that is, the map as it is displayed to the users at runtime).

EXPAND COMMENTS / EXPAND *n*

If you use the option EXPAND COMMENTS, only the initial comment lines of the expand object will be listed; that is, the expand object will be listed until (but not including) the first source-code line which is not a comment line.

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

If you use neither of these two options, the entire expand object will be listed.

expand-type

As *expand-type*, you specify the object type(s) of the expand object(s). The following *expand-types* can be specified:

P	Programs
N	Subprograms
S	External subroutines
H	Helproutines
G	Global data areas
L	Local data areas
A	Parameter data areas
M	Maps
C	Copycodes
E	Error messages
4	Class
*	All object types

If you wish to specify more than one *expand-type*, you can specify them in any sequence and without blanks between them; for example, to have maps, copycodes and subroutines listed within the listed source, specify the *expand-type* as "MCS".

object-name

As *object-name*, you specify the name(s) of the expand object(s) to be listed within the main listed source.

For the *object-name* of an expand object, the same options are available as for the object-name of the main listed object in the primary LIST command syntax. Exceptions: the notations "<" and ">".

Subcommands in Expand Object

Within a listed expand object, only the subcommands PRINT, "+", "- -", and "." are available (see above for explanation of subcommands).

Back to System Commands - Overview.

 Copyright © Software AG 2002

LIST for UNIX/OpenVMS

```
LIST [ [SEQUENTIAL] [object-type] object-name [options]
       DIRECTORY [object-name]
       EXTENDED [extended-type] object-name
       DDM [ddm-name]
       XREF                                      ]
```

object-type

```
{ *
  PROGRAM
  MAP
  DATA - AREAS
    GLOBAL
    LOCAL
    PARAMETER
  ROUTINES
    HELPROUTINE
    SUBPROGRAM|N
    SUBROUTINE|FUNCTION
  COPYCODE
  TEXT
  RECORDING
  DIALOG|3
  CLASS|4
  PROCESSOR|CP
  VIEW                    }
```

The LIST command is used to list one or more objects which are contained in the current library.

Displaying an Individual Source

LIST	If you enter only the LIST command itself, without any parameters, the contents of the source work area will be listed.
LIST *object-name*	If you enter a single *object-name* with the LIST command, you need not specify the *object-type*; the object's source code will be listed.

Displaying Directory Information

LIST DIR	This command displays the directory information about the object currently in the source work area.
LIST DIR *object-name*	This command displays the directory information about the specified object. To display the directory information of several objects, you use asterisk notation for the *object-name*.

LIST *object-type object-name*

If you specify an *object-type*, you must also specify an *object-name*.

To have all objects in the current library listed, you specify "*" for the *object-type*, but no *object-name*.

To have all objects of a certain type listed, you specify a certain *object-type* and "*" for the *object-name*.

If you wish a certain range of objects to be listed, you can use asterisk notation for the *object-name*:

Examples:

LIST * lists all objects in the current library - regardless of their types.

LIST S * lists all subroutines in the current library.

LIST SYS* lists all objects (of any type) whose names begin with "SYS".

LIST M SYS* lists all maps whose names begin with "SYS".

LIST DIR PRG01 lists directory information of object PRG01 in current library.

To select an object from the selection list for a function, you simply mark the object with the appropriate function code in the left-hand column. The function codes are:

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

 Copyright © Software AG 2002

Enter "?" or use F2 to display the list of the available function codes for the selected object.

Back to System Commands - Overview.

LIST for Windows

```
LIST [[object-type] object-name]
```

object-type

```
{ *
  PROGRAM
  MAP
  DATA - AREAS
    GLOBAL
    LOCAL
    PARAMETER
  ROUTINES
    HELPROUTINE
    SUBPROGRAM|N
    SUBROUTINE|FUNCTION
  COPYCODE
  TEXT
  RECORDING
  DIALOG|3
  CLASS|4
  PROCESSOR|CP
  VIEW }
```

The LIST command is used to display objects contained in the current library. When an object is displayed using the LIST command, its content can be copied but not modified.

LIST	If you enter the LIST command without any parameters, the objects marked in the "Objects" or "DDMs" window are displayed.
LIST *object-name*	If you specify the *object-name*, you need not specify the *object-type*; the object's source code is displayed.
LIST*object-type object-name*	If you specify an *object-type*, you also have to specify an *object-name* or an asterisk.
LIST *	To have all objects in the current library listed, specify "*" for the *object-type*, but no *object-name*.
LIST*object-type* *	To have all objects of a certain type listed, specify a certain *object-type* and "*" for the *object-name*.

Asterisk notation can also be used to limit the selection of objects specified in *object-name*:

 Copyright © Software AG 2002

Examples:

LIST * lists all objects in the current library - regardless of their types.

LIST S * lists all subroutines in the current library.

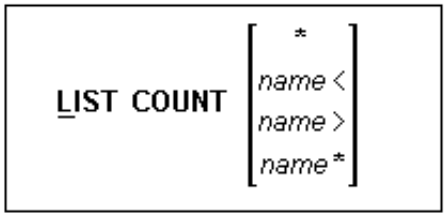
LIST SYS* lists all objects (of any type) whose names begin with "SYS".

LIST M SYS* lists all maps whose names begin with "SYS".

Back to System Commands - Overview.

LIST COUNT

Note:
This command applies to all platforms.



You use the LIST COUNT command to list the number of Natural objects in your current library.

The following command options are available:

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

Note:
If there are objects listed under object type "undefined", this indicates that the library contains objects whose version is not compatible (for example, Natural Version 1.2 maps).

Back to System Commands - Overview.

 Copyright © Software AG 2002

LIST DIR

Note:
This command is for Windows platforms only.

This command displays the directory information about the object currently in the source work area.

LIST DIR *object-name*	This command displays the directory information about the specified object. To display the directory information of several objects, you use asterisk notation for the *object-name*.

Back to System Commands - Overview.

LIST XREF

Note:
This command is for UNIX/OpenVMS and Windows platforms only.

This command displays all active cross-reference data for the current library. This command is only available if Predict with active cross-references is installed.

Note:
This command is only available if Predict has been installed.

Back to System Commands - Overview.

 Copyright © Software AG 2002

LISTSQL

Note:
This command is for Mainframes only.

This command is only available with Natural for DB2 and Natural for SQL/DS.

The command LISTSQL lists those Natural statements in the source code of a programming object which are associated with a database access, and also the corresponding SQL commands they have been translated into.

The LISTSQL command is described in detail in the documentation Natural for DB2 and Natural for SQL/DS.

Back to System Commands - Overview.

LOGOFF

Note:
This command applies to all platforms.

```
LOGOFF
```

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

LOGOFF does **not** cause the Natural session to be terminated. To terminate the session, use the system command FIN,
or execute a program that contains a TERMINATE statement.

For information on LOGOFF processing under Natural Security, see your Natural Security documentation.

LOGOFF has no effect on Natural global parameter settings.

Back to System Commands - Overview.

 Copyright © Software AG 2002

LOGON

Note:
This command applies to all platforms.

```
LOGON library-id [password]
```

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

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

For information on LOGON processing under Natural Security, see your Natural Security documentation.

LOGON causes all Natural global data areas, all assignments made using the SET KEY statement and retained ISN lists to be released. DDMs contained in the DDM buffer area are also released.

Naming Conventions for Libraries

A library ID can be 1 to 8 characters long and must not contain blanks.

A library ID can consist of the following characters:

A - Z	upper-case alphabetical characters
0 - 9	numeric characters
-	hyphen
_	underscore

The first character of a library ID must be an upper-case alphabetical character.

Back to System Commands - Overview.

MAIL

Note:
This command applies to all platforms.

MAIL *mailbox-id*

Note:
This command is available only if Natural Security is installed.

A mailbox is a kind of "notice board" used to broadcast messages, which is defined in Natural Security.

With the MAIL command, you can invoke a mailbox to modify its contents and/or expiration date. The *mailbox-id* you specify (maximum 8 characters) must be defined in Natural Security.

See your Natural Security documentation for details on mailboxes.

Back to System Commands - Overview.

 Copyright © Software AG 2002

MAINMENU

Note:
This command is for Mainframes only. This command is not available via the command line in a remote development environment.

```
MAINMENU [ ON
           OFF
           user-program ]
```

The MAINMENU command may be used to switch ON/OFF Natural main menu mode or to invoke a user program which creates a user-defined menu.

Issuing the MAINMENU command without a parameter has the same effect as MAINMENU ON.

Back to System Commands - Overview.

MAP

Note:
This command applies to Windows platforms and is used for remote development only.

MAP <Server> <Port> [<UserID> [<Password> ['Parm1>=<Value1>;<Parm2>=<Value2>;...']]]

The MAP command enables you to establish a connection to a development server using the command line.

If you enter an asterix as UserID, then the UserID of the client session is used. If you enter an asterix as password, then an empty password string is sent to the development server.

Back to System Commands - Overview.

 Copyright © Software AG 2002

NATLOAD

Note:
This command is for Mainframes and UNIX/OpenVMS platforms only.

```
NATLOAD
```

This command invokes the NATLOAD utility which is contained in the library SYSUNLD.

For information on NATLOAD, see the NATLOAD Utility as described in the Natural NATUNLD/NATLOAD Utilities documentation.

Back to System Commands - Overview.

NATUNLD

Note:
This command is for Mainframes and UNIX/OpenVMS platforms only.

```
NATUNLD
```

This command is used to invoke the NATUNLD utility; see the Natural NATUNLD / NATLOAD Utilities documentation.

Back to System Commands - Overview.

 Copyright © Software AG 2002

NOCOPT

Note:
This command is for Mainframes only.

With this command, you can display or modify the current settings of the Natural Optimizer Compiler options as they were specified during Natural startup.

For more information on NOCOPT, see Switching on the Optimizer Compiler as described in the Natural Optimizer Compiler documentation.

Back to System Commands - Overview.

NOCSHOW

Note:
This command is for Mainframes only.

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

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

Back to System Commands - Overview.

 Copyright © Software AG 2002

NOCSTAT

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

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

Back to System Commands - Overview.

PROFILE

Note:
This command applies to all platforms.

```
PROFILE
```

Note:
This command is available only if Natural Security is installed.

With the PROFILE command, you can display the security profile currently in effect. This profile informs you of the conditions of use in effect for you in your current Natural environment. See also PROFILE Command as described in the Natural Security documentation.

Back to System Commands - Overview.

 Copyright © Software AG 2002

PURGE

Note:
This command is for UNIX/OpenVMS and Windows platforms only.

```
PURGE [object-name] ...
```

The PURGE command is used to delete one or more source objects.

Note:
If the **profile parameter** RECAT is set to ON, the PURGE command will be rejected for a source for which a corresponding cataloged object exists.

Note for UNIX and OpenVMS:
If DDM's reside in a library of the FUSER, they can be deleted with the PURGE command.

As *object-name*, you specify the name(s) of the object(s) to be deleted. You can only delete objects that are stored in the library to which you are currently logged on.

If you wish to delete all objects whose names begin with a specific string of characters, use asterisk notation for the *object-name*.

If you enter the PURGE command without an *object-name*, a list of all objects in the current library will be displayed; on the list, you can then mark the objects to be deleted.

Back to System Commands - Overview.

READ

Note:
This command applies to all platforms.

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

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

object-name

The name of the object to be read. If *object-name* is specified without a library ID, the object will be read only if it is stored in the library to which you are currently logged on.

library-id

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

Under Natural Security, you cannot specify a *library-id*; that is, you can read only objects that are stored in your current library.

Back to System Commands - Overview.

 Copyright © Software AG 2002

REGISTER

Note:
This command applies to all platforms.

```
REGISTER {class–module–name} [ {library-name} [ ES ] ]
         {      *           }  [ {      *     } [ IM ] ]
                                               [ EM ]
```

The system command REGISTER is used to register Natural classes. They are registered for the server ID under which Natural was started.

Note:
On the mainframe, the REGISTER command is only available under TSO and in batch.

For more information on REGISTER, see REGISTER Command as described in the NaturalX documentation.

Back to System Commands - Overview.

RENAME

Note:
This command is for Mainframes only. This command is not available via the command line in a remote development environment.

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

With the RENAME command, you can give a Natural programming object another name.

You can only rename one object at a time. The object to be renamed must be stored in the library to which you are currently logged on.

As *old-name* you specify the existing name of the object to be renamed; as *new-name* you specify the name under which the object is to be stored from now on. To ensure consistency, Natural will rename whatever form(s) of the object exists under the *old-name* (source code or object module or both).

Moreover, when you rename an object, you can also change its object type. This is only possible for objects which exist only in source form. The possible settings you can specify for *new-type* are the same as those you can specify with the editor command SET TYPE (see the sections Program Editor and Data Area Editor).

Back to System Commands - Overview.

 Copyright © Software AG 2002

RENUMBER

Note:
This command applies to all platforms.

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

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

n may be used to specify the increment to be used for renumbering. The default is 10.

Note:
Each time a source is READ into the source area it will be renumbered to the most optimum number space.

Back to System Commands - Overview.

RETURN

Note:
This command applies to all platforms.

```
RETURN [ I  ]
       [ nn ]
       [ *  ]
```

The RETURN command may be used to return to a previous (or initial) Natural application.

If RETURN is specified without any parameters, control will be returned to the previous application (as defined with the system command SETUP). All information about this previous application will be deleted. If no previous application exists, control is returned to the initial application.

If RETURN is issued and no return point is set, the RETURN command will be ignored. Under Natural Security, a LOGOFF command will be executed if RETURN is issued and no return point has been set.

RETURN I	This command causes control to be returned directly to the initial application. This option also causes Natural to delete all definitions of previous applications (except that of the initial application).
RETURN *nn*	This command causes control to be returned to the *nn*th previous application. When this option is used, all information for applications subsequent to the *nn*th application is deleted.
RETURN *	This command will display a list of all return points which are currently set up. On the list you may then select the return point to which you wish to return.

See the SETUP command for further information and examples.

Back to System Commands - Overview.

 Copyright © Software AG 2002

ROUTINES

Note:
This command is for Mainframes only.

```
ROUTINES
```

With this command, you can ascertain which objects in the current library use which external subroutines.

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

If an object is itself a subroutine, the name of the subroutine it contains is displayed.

Back to System Commands - Overview.

RPCERR

Note:
This command is for Mainframes only.

CBT!!!!help id !RPCERR displays the last Natural error number and message if it was RPC related and it also displays the last BROKER reason code and associated message. Additionally, the node and server name from the last Broker call can be retrieved.

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

Back to System Commands - Overview.

 Copyright © Software AG 2002

RUN

Note:
This command applies to all platforms.

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

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

The REPEAT parameter indicates that if the program being executed produces multiple screen output, the screens are to be output one after another without intervening prompting messages. When the program terminates, Natural will enter command mode.

program-name

The name of the program to be run. If no program name is specified, Natural will compile and execute the program currently residing in the source work area.

If *program-name* is specified without a library ID, Natural will read the source program into the source work area (overlaying any existing source program), compile, and execute the specified program only if it is stored under the current library ID. If it is not stored under the current library ID, an error message will be issued.

library-id

The library in which the program to be run is contained. If both *program-name* and *library-id* are specified, Natural will retrieve, compile, and execute the specified program only if it is stored under the library ID specified. If it is not stored under the current library ID, an error message will be issued.

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

A library ID must not be specified if Natural Security is active.

Back to System Commands - Overview.

SAVE

Note:
This command applies to all platforms.

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

The SAVE command is used to store a source object. The contents of the source work area are not affected.

Note:
The SAVE command will be rejected for an object for which a cataloged object module exists and the parameter RECAT=ON is in effect. In this case, use the STOW command to ensure that the contents of source and object code match.

object-name

The name under which the object is to be saved. If a new object is being saved, an object name must be specified. If an object name is not specified, the object will be saved with the name set by the last command which caused a source object to be read into the source work area (for example, EDIT, READ or RUN).

If an object name is specified without a library ID, Natural will determine if the object is stored under the current library ID. If the object is not stored under the current library ID, the object will be saved using the object name specified. Object names within a library must be unique.

library-id

When you save an object under a different name or save a newly created object, the object will, by default, be stored in the current library. If you wish to store it in another library, you have to specify the desired *library-id* after the *object-name*.

Under Natural Security, you cannot specify a *library-id*; that is, you can store an object only in your current library.

Back to System Commands - Overview.

 Copyright © Software AG 2002

SCAN

- For Mainframes
- For UNIX/OpenVMS
- For Windows

SCAN for Mainframes

```
SCAN
```

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

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

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

This section covers the following topics:

- General Rules
- Menu Options
- Direct Commands
- Subcommands
- SCAN under Natural Security

General Rules

- If the library specified is SYSTEM, the library in the FUSER file will be scanned.
- If the name of the specified library begins with SYS but is not SYSTEM, the library in the FNAT file will be scanned.
- The replace option must not be used with maps and data areas.
- It is the user's responsibility to set upper/lower case before invoking the SCAN.
- A scan for a setting that contains lower-case characters or embedded blanks is only possible from the SCAN menu.

Menu Options

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

Field	Input setting
Code	T Statistics Returns the following information: the number of objects that were scanned; the number of objects in which the scan setting was found; the number of source-code lines in which the scan setting was found. L List Modules Containing Scan setting Displays a list of all objects in which the scan setting was found. From the list, you can select individual objects for further processing. S List Scan settings Found Displays one after another each source-code line in which the scan setting was found. See more details below.
Scan setting	The string of characters to be scanned for.
Replace setting	The setting which is to replace the Scan setting.
Library	The ID of the library to be scanned. Default is the current library.
Object Name	The object(s) to be scanned: *blank* all objects ***** all objects *setting*> all objects whose names are greater than or equal to *setting* *setting*< all objects whose names are less than or equal to *setting* If you wish to scan within a certain range of objects, you can use asterisk notation (*) and wildcard notation (?) for the object name, in the same manner as described for the system command LIST.
Selection List	Displays a list of objects from which you can select individual objects (by marking them with any character) for scan processing.
Object Type	You can restrict the search to specific object types. For a selection list of possible types, enter a question mark (?) in this field. If you leave this field blank or enter an asterisk (*), objects of any type will be scanned.
Absolute Scan	If you enter any setting other than **N** or blank, the scan will be "absolute"; that is, the setting to be scanned for will be found in any form, even as part of a longer character string. **Note:** In data areas, a scan is always absolute, regardless of the setting of this parameter.
Trace	**Y** indicates that the trace facility is to be activated. The default is **N**.

 Copyright © Software AG 2002

List Scan settings Found

If you wish, you can modify the lines by using the appropriate SCAN subcommands as described below.

Any object, except maps and data areas, may be modified by the SCAN editor. The object may also be edited using the program editor; this allows the modification of lines other than those containing matches found by the scan. Once the object has been edited, the object should be saved and the editor terminated. Scan processing can then be continued. Before the scan operation invokes another editor you will be prompted with a screen to confirm any updates previously made by the SCAN editor.

Copyright © Software AG 2002

Direct Commands

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

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

Examples of SCAN Command with Keywords:

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

SCAN FUNC=S,SVAL=*setting*,RVAL=*setting*,OBJ=PGM1

Batch Mode

The SCAN command will process only one function per invocation to minimize the repercussions of invalid data being specified. Either keywords or positional parameters may be used. Positional parameters are specified as follows:

SCAN S,*scan-setting*,*replace-setting*,*library*,*object-name*,*object-type*,*absolute*

Note:
To scan for a setting that contains lower-case characters, you do not specify the *scan-setting* in the same line of the batch job as the scan command, but in a separate data line.

 Copyright © Software AG 2002

Subcommands

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

Command	Function
blank	Continue with normal scan processing.
Q	Terminate scan processing.
.	
E	Edit the object using full-screen or line editor.
EDT	
LIST	List the object as it currently appears in the source work area.
LET	Ignore all line changes made after last ENTER.
I	Ignore the object currently being scanned, do not save any modifications, and continue with next object.
.D	Delete line. A **D** will appear next to the line to indicate that it has been deleted.
.L	Ignore any changes after last ENTER. Will also restore any line previously deleted with the line command **.D**.

Editing Rules

- Lines that are marked with **L** cannot be modified.
- If the REPLACE option is used and/or an object is updated in the SCAN editor, the object will always be saved unless an **I, Q** or "**.**" is specified before the next object is scanned.
- Lines containing PASSW, PASSWORD=, CIPHER=, or CIPH= will be ignored by the SCAN command.
- The line length of the source object in the SCAN editor is limited to 72 characters.

SCAN under Natural Security

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

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

Back to System Commands - Overview.

 Copyright © Software AG 2002

SCAN for UNIX/OpenVMS

```
SCAN
```

Note:
The source work area is used by the SCAN command. Therefore, the object currently in the source work area should be SAVEd or STOWed before SCAN is used.

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

When you issue the SCAN command, a window is displayed. In this window, you can specify the following:

Field	Explanation
Scan setting	The setting to be searched for.
Replace setting	The setting which is to replace the scan setting.
Delete setting	Will cause the scan setting to be deleted (YES/NO).
Object Name	The object(s) to be scanned: *blank* or * All objects. *setting** All objects whose names begin with *setting*.
Object Type	The type of object to be scanned: **P** - Programs **C** - Copycodes **N** - Subprograms **S** - Subroutines **H** - Helproutines **M** - Maps **G** - Global data areas **L** - Local data areas **A** - Parameter data areas **T** - Text **4** - Class ***** - objects of all types
Absolute Scan	Will result in an absolute scan (YES/NO). If Absolute Scan is set to "NO", the scan setting must be separated by a blank or specific character.
Library	The ID of the library to be scanned. Default is the current library.
Case Sensitive	**N** - The search will be for the scan setting regardless whether it occurs in upper case, lower case, or a mixture of both. **Y** - The search will be for the scan setting exactly as you specify it.

Make the desired specifications and press ENTER. A window appears displaying the lines containing the searched for scan setting.

```
-----------Search setting BONUS in Program ARRAYD--------
  - Commands:
  - 0070    3 BONUS (1:10)
  - 0340  DISPLAY BONUS (1,1)
  - 0350  DISPLAY BONUS (1:5,1)
  - 0360  DISPLAY BONUS (1:5,1:10)
  - 0380  DISPLAY BONUS (#I:#I + 5)
  - 0390  DISPLAY BONUS (#I:#I - 3)
  - 0400  DISPLAY BONUS (#I+2:J-3)
```

SCAN Subcommands

Any desired SCAN subcommand can be entered. Select "Commands:" and press ENTER. Then you can select one of the following commands:

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

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

SCAN Editing Rules

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

SCAN under Natural Security

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

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

Back to System Commands - Overview.

 Copyright © Software AG 2002

SCAN for Windows

```
SCAN
```

The SCAN command starts a Find Object dialog. With the Find Objects dialog it is possible to find Natural objects and the specified containing text. This dialog is described in detail in the section Introduction to the Natural Studio, paragraph Object Retrieval.

Back to System Commands - Overview.

SCRATCH

Note:
This command is for UNIX/OpenVMS and Windows platforms only.

```
SCRATCH [object-name ...]
```

The SCRATCH command is used to delete one or more objects - in both source and object form.

As *object-name*, you specify the name(s) of the object(s) to be deleted. You can only delete objects which are stored in your current library.

If you wish to delete all objects whose names begin with a specific string of characters, use asterisk notation for the *object-name*.

If you enter the SCRATCH command without an *object-name* or without an *object-name* but with an asterisk, a list of all objects in the current library will be displayed; on the list you may then mark the objects to be deleted.

Notes:
The contents of the source work area is not affected by the SCRATCH command.

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

Back to System Commands - Overview.

 Copyright © Software AG 2002

SETUP

Note:
This command applies to all platforms.

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

The SETUP command is used to define applications to which control is to be returned using the RETURN command. This allows you to easily transfer from one application to another during a Natural session.

If SETUP is issued without any parameters, a menu will be displayed for the purpose of entering the command information.

This section covers the following topics:

- Syntax Explanation
- SETUP/RETURN Example

Syntax Explanation

application-name

The name of the application to which control is to be returned. A maximum of 8 characters may be used (A8).

If *application-name* is blank, a LOGON command will not be issued. This permits multiple return points within the same application.

If *application-name* is "*", the current setting of the system variable *LIBRARY-ID (that is, at the time SETUP is issued) is used to create the LOGON command when RETURN is issued.

command-name

The name of the command which is to be executed when control is returned to the application. A maximum of 60 characters may be used (A60).

If *command-name* is blank, no command will be issued after the LOGON. This is useful for applications under Natural Security for which a startup program has already been defined.

If *command-name* is "*", the current setting of the system variable *STARTUP (that is, at the time SETUP is issued) is used as the startup command when RETURN is issued.

I Option

If the "I" option is specified, all return points defined with previous SETUP commands will be deleted and the application specified with "SETUP I" will be defined as the new initial application.

In a non-Security environment, if you log on from library "SYSTEM" to another library and no return point has been set, this other library will automatically be set as initial return point.

Copyright © Software AG 2002

SETUP/RETURN Example

1. User starts Natural session (default application = "APPL1").
   Return point APPL1 is defined on level 1.
2. User issues command "LOGON APPL2".
3. User executes a program which stacks two commands:
   SETUP *,MENU (establish return point)
   LOGON APPL3 (go to another application)
   Return point APPL2, STARTUP MENU is defined on level 2.
4. User issues command "LOGON APPL4" (user selects another application)
5. User presses a PF key which has the setting "RETURN". Natural will issue for the user:
   LOGON APPL2
   MENU
   Return to APPL2, delete level 2.
6. User executes a program which stacks:
   SETUP *,MENU
   LOGON APPL5
   Return point APPL2, STARTUP MENU is defined on level 2.
7. User executes a program which stacks:
   SETUP *,MENU
   LOGON APPL6
   Return point APPL5, STARTUP MENU is defined on level 3.
8. User executes a program which stacks:
   SETUP *,MENU
   LOGON APPL7
   Return point APPL6, STARTUP MENU is defined on level 4.
9. User executes a program which stacks:
   SETUP *,MENU
   LOGON APPL8
   Return point APPL7, STARTUP MENU is defined on level 5.
10. User executes a program which stacks:
    SETUP *,MENU
    LOGON APPL9
    Return point APPL8, STARTUP MENU is defined on level 6.
11. User issues command "RETURN 2" (return 2 levels back).
    Natural will return user to APPL7, since that was the 2nd previous session (all information for APPL8 is now lost). Level 6 (APPL8) is deleted, level 5 (APPL7) is activated and level deleted.
12. User issues command "RETURN".
    Level 4 (APPL6) is activated, level deleted. Natural will return user to APPL6, since that was the session previous to APPL7.
13. User issues command "RETURN".
    Level 3 (APPL5) is activated, level deleted. Natural will return user to APPL5, since that was the session previous to APPL6.
14. User issues command "RETURN I".
    Level 2 (APPL2) is deleted, level 1 (APPL1) is activated.

Back to System Commands - Overview.

 Copyright © Software AG 2002

SQLERR

Note:
This command is for Mainframes only.

This command is available with Natural for DB2 and Natural for SQL/DS only.

The SQLERR command may be used to obtain information on an SQL error.

The SQLERR command is described in detail in the documentation Natural for DB2 and Natural for SQL/DS.

Back to System Commands - Overview.

STOW

Note:
This command applies to all platforms.

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

The STOW command is used to store an object (in both source and object form) in the Natural system file.

object-name

The name of the object to be stored.

An object name need not be specified if an object is being stored that was read from the system file and is being stored under the same name.

If the object is being stored under a different name, the new object name must be specified.

If a new object is being stored, an object name must be specified. The new object name must not already exist in either source or object form.

library-id

The ID of the library into which the object is to be stored. If both the object name and the library ID are specified, the object name will be stored under the specified library ID. If the object name already exists under the library ID, an error message will be issued.

If Natural Security is active, a library ID cannot be specified.

Notes:
For LDAs, GDAs and Maps the STOW command cannot be invoked from the Next prompt.
If an FDIC system file is specified in the parameter file which is not valid, Natural will display an appropriate error message when the STOW command is issued.

Back to System Commands - Overview.

 Copyright © Software AG 2002

STRUCT

- For Mainframes
- For UNIX/OpenVMS/Windows

STRUCT for Mainframes

```
STRUCT [(n)]
```

The STRUCT command serves two purposes:

- You can use it to perform structural indentation of a source program.
- Various display features make the structure of a program clear to you, thus allowing you to detect any structural inconsistencies.

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

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

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

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

Copyright © Software AG 2002

Generate Structured Source into Work Area

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

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

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

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

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

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

 Copyright © Software AG 2002

Partial Indentation

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

Example of Structural Indentation:

Program before being structurally indented:

```
0010 DEFINE DATA LOCAL
0020 1 EMPL VIEW OF EMPLOYEES
0030 2 PERSONNEL-ID
0040 2 FULL-NAME
0050 3 FIRST-NAME
0060 3 NAME
0070 1 VEHI VIEW OF VEHICLES
0080 2 PERSONNEL-ID
0090 2 MAKE
0100 END-DEFINE
0110 FIND EMPL WITH NAME = 'ADKINSON'
0120 IF NO RECORDS FOUND
0130 WRITE 'NO RECORD FOUND'
0140 END-NOREC
0150 FIND (1) VEHI WITH PERSONNEL-ID = EMPL.PERSONNEL-ID
0160 DISPLAY EMPL.PERSONNEL-ID FULL-NAME MAKE
0170 END-FIND
0180 END-FIND
0190 END
```

The same program after the function Generate Structured Source has been applied to it:

```
0010 DEFINE DATA LOCAL
0020 1 EMPL VIEW OF EMPLOYEES
0030   2 PERSONNEL-ID
0040   2 FULL-NAME
0050     3 FIRST-NAME
0060     3 NAME
0070 1 VEHI VIEW OF VEHICLES
0080   2 PERSONNEL-ID
0090   2 MAKE
0100 END-DEFINE
0110 FIND EMPL WITH NAME = 'ADKINSON'
0120   IF NO RECORDS FOUND
0130     WRITE 'NO RECORD FOUND'
0140   END-NOREC
0150   FIND (1) VEHI WITH PERSONNEL-ID = EMPL.PERSONNEL-ID
0160     DISPLAY EMPL.PERSONNEL-ID FULL-NAME MAKE
0170   END-FIND
0180 END-FIND
0190 END
```

Copyright © Software AG 2002

Display Structure of Source

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

With the Display function, you have the following options:

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

The following information is displayed:

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

 Copyright © Software AG 2002

Example of Display with Structure Information:

```
11:49:23 - Structured Source ABC in Library XYZ -                    2001-01-12
  0010      DEFINE DATA LOCAL                                     *O
  0020      1 EMPL VIEW OF EMPLOYEES                              *O
  0030        2 PERSONNEL-ID                                      *O
  0040        2 FULL-NAME                                         *O
  0050          3 FIRST-NAME                                      *O
  0060          3 NAME                                            *O
  0070      1 VEHI VIEW OF VEHICLES                               *O
  0080        2 PERSONNEL-ID                                      *O
  0090        2 MAKE                                              *O
  0100 0010 END-DEFINE                                            *O
  0110      FIND EMPL WITH NAME = 'ADKINSON'                      *F
  0120        IF NO RECORDS FOUND                                 *FJ
  0130          WRITE 'NO RECORD FOUND'                           *FJ
  0140 0120   END-NOREC                                           *FJ
  0150        FIND (1) VEHI WITH PERSONNEL-ID = EMPL.PERSONNEL-I *FF
  0160          DISPLAY EMPL.PERSONNEL-ID FULL-NAME MAKE          *FF
  0170 0150   END-FIND                                            *FF
  0180 0110 END-FIND                                              *F
  0190      END                                                   *
  PF1=Help, PF2=Menu, PF3=Exit, PF6=Top, PF12=Cancel.
```

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

Print Structure of Source

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

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

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

Write Structure of Source into Work Area

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

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

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

Back to System Commands - Overview.

STRUCT for UNIX/OpenVMS

```
STRUCT [(n)]
```

The STRUCT command performs structural indentation of a source program.

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

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

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

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

Partial Indentation

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

Example of Structural Indentation:

Program before being structurally indented:

```
0010 DEFINE DATA LOCAL
0020 1 EMPL VIEW OF EMPLOYEES
0030 2 PERSONNEL-ID
0040 2 FULL-NAME
0050 3 FIRST-NAME
0060 3 NAME
0070 1 VEHI VIEW OF VEHICLES
0080 2 PERSONNEL-ID
0090 2 MAKE
0100 END-DEFINE
0110 FIND EMPL WITH NAME = 'ADKINSON'
0120 IF NO RECORDS FOUND
0130 WRITE 'NO RECORD FOUND'
0140 END-NOREC
0150 FIND (1) VEHI WITH PERSONNEL-ID = EMPL.PERSONNEL-ID
0160 DISPLAY EMPL.PERSONNEL-ID FULL-NAME MAKE
0170 END-FIND
0180 END-FIND
0190 END
```

The same program after the function Generate Structured Source has been applied to it:

 Copyright © Software AG 2002

```
0010 DEFINE DATA LOCAL
0020 1 EMPL VIEW OF EMPLOYEES
0030   2 PERSONNEL-ID
0040   2 FULL-NAME
0050     3 FIRST-NAME
0060     3 NAME
0070 1 VEHI VIEW OF VEHICLES
0080   2 PERSONNEL-ID
0090   2 MAKE
0100 END-DEFINE
0110 FIND EMPL WITH NAME = 'ADKINSON'
0120   IF NO RECORDS FOUND
0130     WRITE 'NO RECORD FOUND'
0140   END-NOREC
0150   FIND (1) VEHI WITH PERSONNEL-ID = EMPL.PERSONNEL-ID
0160     DISPLAY EMPL.PERSONNEL-ID FULL-NAME MAKE
0170   END-FIND
0180 END-FIND
0190 END
```

The parameter (n) may be supplied to specify the number of spaces used for identation. If (n) is unspecified, identation is set to 2.

Example:

```
STRUCT (5)
```

Back to System Commands - Overview.

SYSADA - ADACALL

Note:
This command is for Mainframes only.

SYSADA invokes the ADACALL utility which is contained in the library SYSADA. The ADACALL utility enables you to issue Adabas direct calls (native commands) directly to an Adabas database from mainframe Natural. The ADACALL utility can be used for learning purposes or for testing/analyzing various problems or scenarios.

For further information on ADACALL, see SYSADA Utility ADACALL as described in the section Debugging and Monitoring of the Utlities for Mainframes documentation.

Back to System Commands - Overview.

 Copyright © Software AG 2002

SYSBPM

Note:
This command is for Mainframes only.

This command invokes the SYSBPM utility. TheSYSBPM utility provides statistical information on the current status of the Natural buffer pool including buffer pool cache and on the objects currently in the buffer pool and buffer pool cache. SYSBPM also offers administration functions.

For further information on SYSBPM, see SYSBPM Utility as described in the section Debugging and Monitoring of the Utilities for Mainframes documentation.

Back to System Commands - Overview.

SYSDDM

Note:
This command is for mainframe and UNIX/OpenVMS platforms only.

This command invokes the SYSDDM utility. The SYSDDM utility is used to perform all functions needed for the creation and maintenance of Natural data definition modules (DDMs).

For further information on the SYSDDM utility, see the section SYSDDM Utility in the mainframe utilities documentation and the section DDM Services in the UNIX/Open VMS User's Guide.

Note for SPoD Users:
This command is not available via the command line in a remote mainframe development environment. This is because DDMs are listed in the tree view under the node DDM and all functions of the SYSDDM utility are available via the context menu or the menu bar.

Back to System Commands - Overview.

 Copyright © Software AG 2002

SYSEDT

Note:
This command is for Mainframes only.

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

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

For further information on SYSEDT, see the sectionSYSEDT Utility as described in the section Debugging and Monitoring in the Utilities documentation.

Back to System Commands - Overview.

SYSERR

Note:
This command applies to all platforms.

```
SYSERR
```

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

For further information on SYSERR, see the Natural SYSERR Utility documentation.

Back to System Commands - Overview.

 Copyright © Software AG 2002

SYSEXT

This command displays the contents of the library SYSEXT which contains various Natural user exits. For each user exit, the following is provided:

- a user exit subprogram (in object form), to include all Natural objects referenced (USRnnnN),
- a program example (in source form) of how to invoke the subprogram (USRnnnP),
- text explaining the function of the user exit (USRnnnT). Each text can be invoked using keywords.

The following functions are provided for each user exit:

- Edit example
- List example
- Run example
- Execute example
- List keywords

- For Mainframes and UNIX/OpenVMS
- For Windows

SYSEXT for Mainframes and UNIX/OpenVMS

```
SYSEXT
```

This command invokes the Natural User Exits Menu. The Natural User Exits Menu displays the contents of the library SYSEXT which contains various Natural user exits. For each user exit, the following is provided:

- a user exit subprogram (in object form), to include all Natural objects referenced (USRnnnN),
- a program example (in source form) of how to invoke the subprogram (USRnnnP),
- text explaining the function of the user exit (USRnnnT). Each text can be invoked using keywords.

Line commands provide the following functions for each user exit:

- Edit example
- List example
- Run example
- Execute example
- List keywords

Keywords

Keywords help you find the user exits relevant to your current task.

To get a list of the keywords relevant to the current user exit:

- Issue the K line command.

To display information on a user exit:

1. Enter a keyword relevant to the user exit in the Keyword field.
2. Enter the D line command next to the text you wish to display.

Back to System Commands - Overview.

 Copyright © Software AG 2002

SYSEXT for Windows

SYSEXT

The SYSEXT command invokes the User Exit function of the Natural SYSEXT utility. The User Exit dialog displays the contents of the library SYSEXT which contains various Natural user exits. For each user exit, the following is provided:

- a user exit subprogram (in object form), to include all Natural objects referenced (USRnnnN),
- a program example (in source form) of how to invoke the subprogram (USRnnnP),
- text explaining the function of the user exit (USRnnnT). Each text can be invoked using keywords.

For each listed user exit you can perform the functions:

- Edit example
- List example
- Run example
- Execute example
- List documentation
- List Keywords

Keywords help you find the user exits relevant to your current task.You can list the available keywords either by user exit or by keyword. You can also list the keywords relevant to the selected user exit.

In the (nameless) list box at bottom of screen, text members listed under User Exit.

Do all dialog elements need a name?

Back to System Commands - Overview.

SYSFILE

- For Mainframes
- For UNIX/OpenVMS and Windows

SYSFILE for Mainframes

```
SYSFILE [ WORKFILE ]
        [ PRINTER  ]
```

This command is used to invoke the function Natural Print/Work Files of the Natural SYSFILE utility.

This function provides information on the work files and print files available.

If you enter only the command SYSFILE itself, work file **and** print file assignments are displayed.

If you enter SYSFILE WORKFILE, only the work file assignments are displayed.

If you enter SYSFILE PRINTER, only the print file assignments are displayed.

For further information, see the section SYSFILE under Debugging and Monitoring in the Natural Utilities for Mainframes documentation.

Back to System Commands - Overview.

 Copyright © Software AG 2002

SYSFILE for UNIX/OpenVMS and Windows

```
SYSFILE [ WORKFILE ]
        [ PRINTER  ]
```

This command invokes the Work and Print Files function of the Natural SYSFILE utility. With SYSFILE you can obtain information about the following:

- Reports
- Logical Devices
- Defined Physical Devices
- Defined Printer Profiles
- Defined Workfiles

Back to System Commands - Overview.

SYSMAIN

Note:
This command applies to all platforms.

```
SYSMAIN
```

This command invokes the SYSMAIN utility. The SYSMAIN utility is used to perform object operations in Natural such as copy, move and delete. The SYSMAIN utility is also used to transfer objects within the Natural system from one environment to another using the import function.

Mainframe: See the section SYSMAIN Utility - Overview in the Utilities documentation.

Windows: See the section The SYSMAIN Utility in the User's Guide.

UNIX/Open VMS: See the section SYSMAIN Utility in the User's Guide.

Back to System Commands - Overview.

 Copyright © Software AG 2002

SYSNCP

Note:
This command applies to all platforms.

```
SYSNCP
```

This command invokes the SYSNCP utility.

For information on SYSNCP, see the Natural SYSNCP Utility documentation.

Back to System Commands - Overview.

SYSOBJH

Note:
This command is for UNIX/OpenVMS and Windows platforms only.

```
SYSOBJH
```

This command is used to invoke the SYSOBJH utility.
For further information, see the SYSOBJH Utility documentation under Natural Tools/All Platforms.

Back to System Commands - Overview.

 Copyright © Software AG 2002

SYSPARM

Note:
This command is for Mainframes only.

This command invokes the SYSPARM utility which is used to create and maintain strings of Natural profile parameters. You can specify a string of parameters and store it under a profile name. You can then then invoke Natural with the single dynamic parameter PROFILE=*profile-name*. The parameters defined in this profile are then passed to Natural as dynamic parameters.

For further information on SYSPARM, see SYSPARM Utility as described in the Natural Utilities for Mainframes documentation.

Back to System Commands - Overview.

SYSPAUL

Note:
This command applies to Windows platforms only.

```
SYSPAUL
```

This command invokes the SYSPAUL utility.

Please, note that the SYSPAUL utility is going to be withdrawn soon. Its functionality will be covered by the Natural utility SYSOBJH.

The Natural utility SYSPAUL allows you to define application descriptions, to unload and package (for example, on diskette) entire Natural applications, and to load and scan individual application files.

This utility is described in the Natural Operations documentation.

Back to System Commands - Overview.

 Copyright © Software AG 2002

SYSPROD

- For Mainframes and UNIX/OpenVMS
- For Windows

SYSPROD for Mainframes and UNIX/OpenVMS

```
SYSPROD
```

With the SYSPROD command, you can ascertain which products are installed at your Natural site. You are given information on your current Natural version, Natural Selectable Units and products running with or under Natural.

When you enter the command, a window is displayed, listing the following information for each product installed:

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

For some of the products listed, you can get additional information by marking them with a command. For a list of available commands, enter a "?" in the "Cmd" column of the SYSPROD window.

Back to System Commands - Overview.

SYSPROD for Windows

```
SYSPROD
```

With the SYSPROD command, you can ascertain which products are installed at your Natural site. You are given information on your current Natural version, Natural Selectable Units and products running with or under Natural.

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

- the product identification code (ID),
- the product name,
- the product version,
- the patch level,
- the installation date and time.

You can obtain more detailed information on any of the products listed by selecting "View > Detailed View" in the menu bar.

Back to System Commands - Overview.

 Copyright © Software AG 2002

SYSPROF

- For Mainframes
- For UNIX/OpenVMS
- For Windows

SYSPROF for Mainframes

```
SYSPROF
```

With the SYSPROF command, you can have the current definitions of the Natural system files displayed.

For each system file, the database ID, file number and database type are displayed.

Back to System Commands - Overview.

SYSPROF for UNIX/OpenVMS

```
SYSPROF
```

With the SYSPROF command, you can display the current definitions of the Natural system files. For each file, you can display:

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

Back to System Commands - Overview.

 Copyright © Software AG 2002

SYSPROF for Windows

```
SYSPROF
```

With the SYSPROF command, you can display the current definitions of the Natural system files. For each file, you can display:

- the file name,
- the database ID,
- the file number,
- the database type,
- the path, if in File System,
- logical file number if assigned.

Using the View box, you can view system files only, files in the file system only or all files.

Back to System Commands - Overview.

SYSRPC

Note:
This command applies to all platforms.

```
SYSRPC
```

This command is used to invoke the SYSRPC utility. The Natural SYSRPC utility provides functions for maintaining remote procedure calls. This utility is described in the SYSRPC Utility documentation. For information on how to apply the SYSRPC utility functions to establish a framework for communication between server and client systems, refer to the Natural RPC (Remote Procedure Call) documentation.

Back to System Commands - Overview.

 Copyright © Software AG 2002

SYSTP

Note:
This command is for Mainframes only.

This command invokes the SYSTP utility whichallows you to monitor and control various TP-monitor-specific characteristics of Natural.

For information on SYSTP, see the sectionSYSTP Utility as described in the Utilities documentation under Debugging and Monitoring.

Back to System Commands - Overview.

SYSTRANS

Note:
This command applies to all platforms.

```
SYSTRANS
```

This command is used to invoke the SYSTRANS utility. You use SYSTRANS to transfer all Natural objects, maps, DDMs, libraries, command processors and error messages as well as Adabas FDTs from one hardware platform to another.

This utility is described in the Natural SYSTRANS Utility documentation.

Back to System Commands - Overview.

 Copyright © Software AG 2002

SYSUNLD

Note:
This command is for Mainframe and UNIX/OpenVMS platforms only.

```
SYSUNLD
```

This command is used to invoke the library SYSUNLD. SYSUNLD contains the utilities NATUNLD and NATLOAD which are used to transfer Natural programming objects, error messages and DDMs between system files and work files.

These utilities are described in the Natural NATUNLD/NATLOAD Utilities documentation.

Back to System Commands - Overview.

TECH

Note:
This command applies to all platforms.

TECH

This command displays the following technical and other information about your Natural session:

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

Note for Mainframes, UNIX/OpenVMS and Windows in character applications only:
To display this information from any point in an application, you can use the terminal command %<TECH. In addition the following information is still available under Windows: Names, types and levels of the currently active programming object and all objects on higher levels.

Note for Windows platforms:
This command is also available in a remote session.
All information can be read in batch mode.

Back to System Commands - Overview.

 Copyright © Software AG 2002

TEST

Note:
This command is for Mainframes only.

```
TEST [ON]
     [OFF]
```

With the TEST command, you invoke the Natural Debugger which is one of the Natural Test Utilities.

If you enter the command TEST without parameters, the main menu of the Natural Debugger is displayed. With TEST ON/OFF, you activate/de-activate the test mode of the Debugger.

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

For information on the Natural Debugger, see theNatural Debugger documentation in the section Debugging and Monitoring of the Natural Utilities documentation. This section also contains information on other Natural utilities used for online testing and monitoring.

Back to System Commands - Overview.

UNCATALOG

Note:
This command is for UNIX/OpenVMS and Windows platforms only.

```
UNCATALOG [object-name]...
```

The UNCATALOG command is used to delete one or more object modules.

As *object-name*, you specify the name of the object to be deleted. You can only delete objects which are stored in the library to which you are currently logged on.

If more than one object is to be deleted, the *object-names* must be separated by one or more blanks (or the currently defined delimiter character).

If you wish to delete all objects whose names begin with a specific string of characters, use asterisk notation for the *object-name*.

If you enter the UNCATALOG command without an *object-name* or without an *object-name* but with an asterisk, a list of all cataloged objects in the current library will be displayed; on the list, you can then mark the object(s) to be deleted.

The contents of the source work area is not affected by the UNCATALOG command.

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

Back to System Commands - Overview.

 Copyright © Software AG 2002

UNLOCK

Note:
This command applies to Windows platforms where it is used for remote development only. It enables you to view locked objects and unlock them if need be.

This command is recommended for use by the Natural administrator only. However, the administrator can enable the use of this command for each user profile in Natural Security.

For further information refer to the topicObject Locking in the Windows remote Development documentation.

To display locked objects

1. From the Menu Bar, select Tools, choose Development Tools and select Unlock Objects.
   Or type UNLOCK in the command line and press Enter.
2. In the resulting dialog box, you can enter the following selection criteria:
   - Application name
     As a special input, you can enter a blank character to find all locked objects, irrespective of whether they are linked to an application or not.
   - Library
   - DBID (database ID)
   - FNR (file number)
   - Object name
   - Object type
3. The search results can be narrowed by entering the additional selection criteria "Locked user ID" and/or "Locked date".
4. Click the OK button to confirm your entries.

If you entered the name of a locked object in the field "Object name", this object is unlocked immediately and a corresponding message is displayed.

If you did not enter a specific object name, the locked objects found are listed in a results window.

To unlock objects in the results window

1. In the results window, select the object(s) to be unlocked.
2. Click the right mouse button and, from the resulting context menu, choose Unlock Objects.

To delete the results window

1. Select any object in the results window.
2. Click the right mouse button and, from the resulting context menu, choose Delete Tab.

Back to System Commands - Overview.

UNMAP

Note:
This command applies to Windows platforms where it is used for remote development only.

UNMAP

You can use the UNMAP command to disconnect a session on a mainframe server. When issued in the command line, the UNMAP command disconnects the currently active remote environment.

Back to System Commands - Overview.

 Copyright © Software AG 2002

UNREGISTER

Note:
This command applies to all platforms.

This command is used to unregister Natural classes.

See the UNREGISTER Command in the NaturalX documentation.

Back to System Commands - Overview.

UPDATE

Note:
This command applies to all platforms.

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

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

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

When the system command CHECK is used with UPDATE OFF, an error message is displayed.

The UPDATE command has no effect on other Natural system commands.

Back to System Commands - Overview.

 Copyright © Software AG 2002

XREF

Note:
This command applies to all platforms.

```
        [ ON    ]
        [ OFF   ]
XREF    [ FORCE ]
        [ DOC   ]
        [ ?     ]
```

Note:
This command is only available if Predict has been installed.

This command controls the usage of the Predict function "active cross-references".

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

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

To look at cross-reference data, you use the XREF option of the system command LIST.

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

The following command options are available:

XREF	If you enter the XREF command without parameters, a menu/dialog is displayed where you specify the desired option.
XREF ON	This command activates the active cross-reference function. Cross-reference data will be stored in the respective Predict entries each time a Natural program/data area is cataloged.
XREF OFF	This command deactivates the active cross-reference facility. No cross-reference data will be stored. Existing cross-reference data for the object being cataloged will be deleted.
XREF FORCE	The object can only be cataloged if a Predict entry exists for it. When the object is cataloged, its cross-reference data will be stored in Predict. If no Predict entry exists, the object cannot be cataloged.
XREF DOC	The object can only be cataloged if a Predict entry exists for it. However, when the object is cataloged, no cross-reference data will be stored in Predict, and existing cross-reference data for the object will be deleted. If no Predict entry exists, the object cannot be cataloged.
XREF ?	**For Mainframes and UNIX/OpenVMS only:** With "XREF ?" you can call the Help function for the XREF command.

Natural Security Considerations

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

Back to System Commands - Overview.

 Copyright © Software AG 2002

Programming Reference

This documentation applies to all platforms on which Natural can be used. It contains detailed information which you will find necessary when writing Natural applications.

General Information	General information on the various topics related to the usage of Natural statements: user-defined variables and constants, report specification, text notation, logical condition criteria, and rules for arithmetic assignment.
System Variables	Describes the Natural system variables. You may use system variables to obtain various types of system information (for example, the current date and time).
Large and Dynamic Variables/Fields	As of Natural Version 4.1 for Windows, Unix and OpenVMS, enhanced capabilities are provided for the usage of large variables by removing the existing size limitations and by providing for dynamic allocation of these variables at execution time.
System Functions	Describes the Natural system functions. You may use system functions to obtain various statistical and mathematical results.
Terminal Commands	Describes the Natural terminal commands. These commands generally perform some terminal-related activities.
Keywords and Reserved Words	List of all Natural keywords and reserved words.

Syntax Symbols

In the diagrams which describe the syntax of Natural statements and parts of statements, several symbols are used. These symbols are explained in the Natural Statements documentation.

General Information

This page covers the following topics related to the usage of Natural statements:

- User-Defined Variables
- Constants
- Report Specification
- Text Notation
- User Comments
- End of a Statement
- Logical Condition Criteria
- Rules for Arithmetic Assignment

User-Defined Variables

User-defined variables can be used to store intermediate results in a program or routine.

- Naming Conventions
- Definition of Variables
- Statement Reference Notation - r
- Definition of Format and Length
- Special Formats
- Index Notation
- Referencing a Database Array
- Referencing the Internal Count for a Database Array
- Qualifying Data Structures

Naming Conventions

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

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

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

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

 Copyright © Software AG 2002

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

Character	Explanation
A - Z	alphabetical characters (upper and lower case)
0 - 9	numeric characters
-	hyphen
@	at sign
_	underline
/	slash
$	dollar sign
§	paragraph sign
&	ampersand
#	hash/number sign
+	plus sign (only allowed as first character)

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

- an upper-case alphabetical character
- #
- +
- &

If the first character is a "#", "+" or "&", the name must consist of at least one additional character.

"+" as the first character of a name is only allowed for application-independent variables (AIVs) and variables in a global data area. Names of AIVs must begin with a "+".

"&" as the first character of a name is used in conjunction with dynamic source program modification (see the RUN statement in the Natural Statements documentation), and as a dynamically replaceable character when defining processing rules (see the map editor description in your Natural User's Guide).

Definition of Variables

You define the characteristics of a variable with the following notation:

(r,format-length/index)

This notation follows the variable name, optionally separated by one or more blanks. No blanks are allowed between the individual elements of the notation. The individual elements may be specified selectively as required, but when used together, they must be separated by the characters as indicated above.

Attention:
If operating in structured mode or if a program contains a DEFINE DATA LOCAL clause, variables cannot be defined dynamically in a statement. This does not apply to application-independent variables (AIVs).

Statement Reference Notation - r

A statement label or the source-code line number can be used to refer to a previous Natural statement. This can be used to override Natural's default referencing (as described for each statement, where applicable), or for documentation purposes.

Default Referencing of Database Fields

Generally, the following applies if you specify no statement reference notation: By default, the innermost active database loop (FIND, READ or HISTOGRAM) in which the database field in question has been read is referenced. If the field is not read in any active database loop, the last previous GET statement (in reporting mode also FIND FIRST or FIND UNIQUE statement) which has read the field is referenced.

Referencing with Statement Labels

Any Natural statement which causes a processing loop to be initiated and/or causes data elements to be accessed in the database may be marked with a symbolic label for subsequent referencing.

A label may be specified either in the form label. before the referencing object or in parentheses (label.) after the referencing object (but not both simultaneously).

The naming conventions for labels are identical to those for variables. The period after the label name serves to identify the entry as a label.

Example:

```
... RD. READ PERSON-VIEW BY NAME STARTING FROM 'JONES'   FD. FIND AUTO-VIEW WITH PERSONNEL-ID = PERSONNEL-ID (FD.)        DISPLAY NAME (RD.) FIRST-NAME (RD.) MAKE (FD.)
    END-FIND END-READ ...
```

Referencing with Source-Code Line Numbers

A statement may also be referenced by using the number of the source-code line in which the statement is located.

All four digits of the line number must be specified (leading zeros must not be omitted).

Example:

```
... 0110 FIND EMPLOYEES-VIEW WITH NAME = 'SMITH' 0120   FIND VEHICLES-VIEW WITH MODEL = 'FORD' 0130     DISPLAY NAME (0110) MODEL (0120) 0140   END-FIND 0150 END-FIND ...
```

For further information on the referencing of statements, see the Natural Programming Guide.

 Copyright © Software AG 2002

Definition of Format and Length

Fixed-length variables can be defined with the following formats and corresponding lengths:

Note:
For the definition of Format and Length in dynamic variables, see Definition of Dynamic Variables.

Format		Definable Length (Number of Digits)	Internal Length (in Bytes)
A	Alphanumeric - on mainframe computers: - on all other platforms:	**1 - 253** **1 - 1073741824**	1 - 253 1 - 1073741824
B	Binary - on mainframe computers: - on all other platforms:	**1 - 126** **1 - 1000000000**	1 - 126 1 - 1000000000
C	Attribute Control	-	2
D	Date	-	4
F	Floating Point	**4** or **8**	4 or 8
I	Integer	**1**, **2** or **4**	1, 2 or 4
L	Logical	-	1
N	Numeric (unpacked)	**1 - 29**	1 - 29
P	Packed numeric	**1 - 29**	1 - 15
T	Time	-	7

Length can only be specified if format is specified. With some formats, the length need not be explicitly specified (as shown in the table above).

For fields defined with format N or P, you can use decimal position notation in the form "nn.m". "nn" represents the number of positions before the decimal point, and "m" represents the number of positions after the decimal point. The sum of the values of "nn" and "m" must not exceed 29 and the value of "m" must not exceed 7.

Note:
In reporting mode, if format and length are not specified for a user-defined variable, the default format/length N7 will be used, unless this default assignment has been disabled by the the session parameter FS.

For a database field, the format/length as defined for the field in the DDM apply. (In reporting mode, it is also possible to define in a program a different format/length for a database field.)

In structured mode, format and length may only be specified in a data area definition or with a DEFINE DATA statement.

Example of Format/Length Definition - Structured Mode:

```
DEFINE DATA LOCAL 1 EMPLOY-VIEW VIEW OF EMPLOYEES   2 NAME   2 FIRST-NAME 1 #NEW-SALARY (N6.2) END-DEFINE ... FIND EMPLOY-VIEW ... ... COMPUTE #NEW-SALARY = ... ...
```

In reporting mode, format/length may be defined within the body of the program, if no DEFINE DATA statement is used.

Example of Format/Length Definition - Reporting Mode:

```
...
 FIND EMPLOYEES... ... COMPUTE #NEW-SALARY(N6.2) = ......
```

Special Formats

In addition to the standard alphanumeric (A) and numeric (B, F, I, N, P) formats, Natural supports the special formats C, D, T and L, which are described below.

Format C - Attribute Control

A variable defined with format C may be used to assign attributes dynamically to a field used in a DISPLAY, INPUT or WRITE statement.

For a variable of format C, no length can be specified. The variable is always assigned a length of 2 bytes by Natural.

Example:

```
 DEFINE DATA LOCAL 1 #ATTR(C) 1 #A(N5) END-DEFINE ... MOVE (AD=I CD=RE) TO #ATTR INPUT #A (CV=#ATTR) ...
```

For further information, see the session parameter CV.

Formats D - Date, and T - Time

Variables defined with formats D and T can be used for date and time arithmetic and display. Format D can contain date information only. Format T can contain date and time information; in other words, date information is a subset of time information. Time is counted in tenths of seconds.

For variables of formats D and T, no length can be specified. A variable with format D is always assigned a length of 4 bytes (P6) and a variable with format T is always assigned a length of 7 bytes (P12) by Natural.

Example:

```
 DEFINE DATA LOCAL 1 #DAT1 (D)
  END-DEFINE * MOVE *DATX TO #DAT1 ADD 7 TO #DAT1 WRITE '=' #DAT1 END
```

For further information, see the session parameter EM and the system variables *DATX and *TIMX.

The value in a date field must be in the range from 1st January 1582 to 31st December 2699.

Format L - Logical

A variable defined with format L may be used as a logical condition criterion. It can take the value "TRUE" or "FALSE".

For a variable of format L, no length can be specified. A variable of format L is always assigned a length of 1 byte by Natural.

Example:

```
 DEFINE DATA LOCAL 1 #SWITCH(L) END-DEFINE MOVE TRUE TO #SWITCH ... IF #SWITCH    ...    MOVE FALSE TO #SWITCH ELSE    ...    MOVE TRUE TO #SWITCH END-IF
```

 Copyright © Software AG 2002

For further information on logical value presentation, see the session parameter EM.

Format "Handle"

A variable defined as "HANDLE OF dialog-element-type" can be used as a GUI handle.

A variable defined as "HANDLE OF OBJECT" can be used as an object handle.

For further information on GUI handles, see the (WIN::WIN]Natural User's Guide for Windows. For further information on object handles, see the NaturalX documentation.

Index Notation

An index notation is used for fields that represent an array.

An integer numeric constant or user-defined variable may be used in index notations. A system variable, system function or qualified variable cannot be used in index notations.

Array Definition - Examples:

1. **#ARRAY (3)**
   Defines a one-dimensional array with three occurrences.
2. **FIELD (*label*.,A20/5) or *label*.FIELD(A20/5)**
   Defines an array from a database field referencing the statement marked by "*label*." with format alphanumeric, length 20 and 5 occurrences.
3. **#ARRAY (N7.2/1:5,10:12,1:4)**
   Defines an array with format/length N7.2 and three array dimensions with 5 occurrences in the first, 3 occurrences in the second and 4 occurrences in the third dimension.
4. **FIELD (*label*./i:i + 5) or *label*.FIELD(i:i + 5)**
   Defines an array from a database field referencing the statement marked by "*label*.". FIELD represents a multiple-value field or a field from a periodic group where "*i*" specifies the offset index within the database occurrence. The size of the array within the program is defined as 6 occurrences (i:i + 5). The database offset index is specified as a variable to allow for the positioning of the program array within the occurrences of the multiple-value field or periodic group. For any repositioning of "*i*" a new access must be made to the database via a GET or GET SAME statement.

Natural allows for the definition of arrays where the index does not have to begin with "1". At runtime, Natural checks that index values specified in the reference do not exceed the maximum size of dimensions as specified in the definition.

Note:
For compatibility with Natural Version 1, an array range may be specified using a hyphen (-) instead of a colon (:). A mix of both notations, however, is not permitted. The hyphen notation is only allowed in reporting mode (but not in a DEFINE DATA statement).

On mainframe computers, index values may be in the range from -32767 to +32767. The maximum number of occurrences per array is 32767. The maximum size of an entire array is 32767 bytes (= 32 KB - 1). The maximum size of a data area per programming object is 16,777,215 bytes (16 MB - 1).

On all other platforms, the maximum index value is 1,073,741,824. The maximum size of a data area per programming object is 1,073,741,824 bytes (1 GB). Use the DSLM profile parameter to reduce these limits for compatibility reasons to the limits applicable for mainframe computers.

Simple arithmetic expressions using the "+" and "-" operators may be used in index references. When arithmetic expressions are used as indices, the operators "+" or "-" must be preceded and followed by a blank.

Arrays in group structures are resolved by Natural field by field, not group occurrence by group occurrence.

Example of Group Array Resolution:

```
DEFINE DATA LOCAL 1 #GROUP (1:2)   2 #FIELDA (A5/1:2)   2 #FIELDB (A5) END-DEFINE ...
```

If the group defined above were output in a WRITE statement:

```
WRITE #GROUP (*)
```

the occurrences would be output in the following order:

#FIELDA(1,1) #FIELDA(1,2) #FIELDA(2,1) #FIELDA(2,2) #FIELDB(1) #FIELDB(2)

and `not`:

```
#FIELDA(1,1) #FIELDA(1,2) #FIELDB(1) #FIELDA(2,1) #FIELDA(2,2) #FIELDB(2)
```

Array Referencing - Examples:

1. **#ARRAY (1)**
   References the first occurrence of a one-dimensional array.
2. **#ARRAY (7:12)**
   References the seventh to twelfth occurrence of a one-dimensional array.
3. **#ARRAY (i + 5)**
   References the i+fifth occurrence of a one-dimensional array.
4. **#ARRAY (5,3:7,1:4)**
   Reference is made within a three dimensional array to occurrence 5 in the first dimension, occurrences 3 to 7 (5 occurrences) in the second dimension and 1 to 4 (4 occurrences) in the third dimension.
5. An asterisk may be used to reference all occurrences within a dimension:
   ```
   DEFINE DATA LOCAL
   1 #ARRAY1 (N5/1:4,1:4)
   1 #ARRAY2 (N5/1:4,1:4)
   END-DEFINE
   ...
   ADD #ARRAY1 (2,*) TO #ARRAY2 (4,*)
   ...
   ```

Using a Slash before an Array Occurrence

If a variable name is followed by a 4-digit number enclosed in parentheses, Natural interprets this number as a line-number reference to a statement. Therefore a 4-digit array occurrence must be preceded by a slash "/" to indicate that it is an array occurrence; for example:

```
#ARRAY(/1000) not: #ARRAY(1000)
```

because the latter would be interpreted as a reference to source code line 1000.

If an index variable name could be misinterpreted as a format/length specification, a slash "/" must be used to indicate that an index is being specified. If, for example, the occurrence of an array is defined by the value of the variable "N7", the occurrence must be specified as:

```
#ARRAY (/N7) not: #ARRAY (N7)
```

 Copyright © Software AG 2002

because the latter would be misinterpreted as the definition of a 7-byte numeric field.

Referencing a Database Array

Referencing Multiple-Value Fields and Periodic-Group Fields

A multiple-value field or periodic-group field within a view/DDM may be defined and referenced using various index notations.

For example, the first to tenth values and the Ith to Ith+10 values of the same multiple-value field/periodic-group field of a database record:

```
DEFINE DATA LOCAL 1 I(I2) 1 EMPLOY-VIEW VIEW OF EMPLOYEES   2 LANG (1:10)   2 LANG (I:I + 10) END-DEFINE
```

or:

```
RESET I(I2) ... READ EMPLOYEES OBTAIN LANG(1:10) LANG(I:I + 10)
```

Note:
The same lower bound index may only be used once per array, (this applies to constant indexes as well as variable indexes). For an array definition using a variable index, the lower bound must be specified using the variable by itself, and the upper bound must be specified using the same variable plus a constant.

Referencing Arrays defined with Constants

An array defined with constants may be referenced using either constants or variables. The upper bound of the array cannot be exceeded. The upper bound will be checked by Natural at compilation time if a constant is used.

```
RESET I(I2) I = 1 READ EMPLOYEES OBTAIN LANG(1:10) WRITE LANG(1) / LANG(5:9) / LANG(1:10)
```

```
DEFINE DATA LOCAL 1 I(I2) 1 EMPLOY-VIEW VIEW OF EMPLOYEES   2 LANG (1:10) END-DEFINE *
READ EMPLOY-VIEW   FOR I 1 TO 5     WRITE LANG(1.I)
  END-FOR END-READ END
```

If a multiple-value field or periodic-group field is defined several times using constants and is to be referenced using variables, the following syntax is used:

```
DEFINE DATA LOCAL 1 I(I2)
 1 J(I2)
 1 EMPLOY-VIEW VIEW OF EMPLOYEES   2 LANG (1:5)   2 LANG (11:20) END-DEFINE *
 READ EMPLOY-VIEW   FOR I 1 TO 2     FOR J I TO 4       DISPLAY 'LANGUAGE' LANG(1.I:J)     END-FOR   END-FOR END-READ END
```

Referencing Arrays defined with Variables

Multiple-value fields or periodic-group fields in arrays defined with variables must be referenced using the same variable.

```
RESET I(I2) I = 1 READ EMPLOYEES OBTAIN LANG(I:I+10) WRITE LANG(I) / LANG (I+5:I+6) END
```

If a different index is to be used, an unambiguous reference to the first encountered definition of the array with variable index must be made. This is done by qualifying the index expression as shown below:

```
RESET I(I2) J(I2) I = 1 J = 1 READ EMPLOYEES OBTAIN LANG(I:I+10) WRITE LANG(I.J) / LANG(I.1:5) END
```

The expression "I." is used to create an unambiguous reference to the array definition and "positions" to the first value within the read array range (LANG(I: I + 10)).

The current content of "I" at the time of the database access determines the starting occurrence of the database array.

Referencing Multiple-Defined Arrays

For multiple-defined arrays, a reference with qualification of the index expression is usually necessary to ensure an unambiguous reference to the desired array range.

Example:

```
DEFINE DATA LOCAL   1 I(I2) INIT <1>   1 J(I2) INIT <2>   1 EMPLOY-VIEW VIEW OF EMPLOYEES
      2 LANG (1:10)
      2 LANG (5:20)END-DEFINEREAD (2) EMPLOY-VIEW  WRITE LANG(1.1:10) / LANG(5.5:15)  DISPLAY LANG(1.I:I+2) / LANG(5.J)END-READEND
```

A similar syntax is also used if multiple-value fields or periodic-group fields are defined using index variables.

Example:

```
DEFINE DATA LOCAL  1 I(I2) INIT <1>  1 J(I2) INIT <1>  1 N(I2) INIT <1>  1 M(I2) INIT <1>  1 EMPLOY-VIEW VIEW OF EMPLOYEES    2 LANG (I:I+10)    2 LANG (J:J+5)    2 LANG (1:2) END-DEFINE READ (2) EMPLOY-VIEW  WRITE LANG(I.I) / LANG(I.I:I+10)  DISPLAY LANG(J.N) / LANG(J.N:M)  DISPLAY LANG(1.N) / LANG(1.N:M) END-READ END
```

Referencing the Internal Count for a Database Array

It is sometimes necessary to reference a multiple-value field and/or a periodic group without knowing how many values/occurrences exist in a given record. Adabas maintains an internal count of the number of values of each multiple-value field and the number of occurrences of each periodic group. This count may be referenced by specifying "C*" immediately before the field name. See also the data-area-editor line command ".*" (as described in your Natural User’s Guide).

The count is returned in format N3.

Examples:

C*LANG	Returns the count of the number of values for the multiple-value field LANG.
C*INCOME	Returns the count of the number of occurrences for the periodic group INCOME.
C*BONUS(1)	Returns the count of the number of values for the multiple-value field BONUS in periodic group occurrence 1 (assuming that BONUS is a multiple-value field within a periodic group.)

Note for SQL databases:
The C* notation cannot be used for SQL databases.

Note for VSAM databases:
The C* notation does not return the number of values/occurrences but the maximum occurrence/value as defined in the DDM (MAXOCC).

```
Example of C* Function:
* EXAMPLE 'COUNT' /* USING C*NOTATION TO OBTAIN INTERNAL COUNT FOR DATABASE ARRAY /*************************************** LOOP1 / READ ADDRESSES BY EXT1 BEFORE VALUE(1:5) WHITE NOTICE 'NAME' 'CODE' / *NUMBER IN LANGUAGES DURING  #*LANG                    GET *LANGUAGE(*) / LANG (*)  GET *LANGUAGE(*) / LANG (*)  ********************************** WRITE 'VALUES' NOTE / FOR IN (1:5) START 1 TO #*INCOME  WRITE 'VALUES' GET VALUES(*) (1:5) LOOP  ********************************** WRITE 'TOTAL YEAR BONUS' #*BONUS(1) BONUS (1,1) BONUS (1,2)  /* COUNT YEAR BONUS: #*BONUS(2) BONUS (2,1) BONUS (2,2) NOTE 1,2
END /* PROCESSING OF VARIOUS VARIABLES  /* VARIABLE 1,2,3,4  /* VARIABLE 5,6,7,8  /* VARIABLE 9,10  /* VARIABLE 11,12  /* VARIABLE 13,14,15,16  /* VARIABLE 17,18,19,20  /* VARIABLE 21,22,23,24  /* VARIABLE 25,26,27,28  /* VARIABLE 29,30,31,32  /* VARIABLE 33,34,35,36  /* VARIABLE 37,38,39,40  /* VARIABLE 41,42,43,44  /* VARIABLE 45,46,47,48  /* VARIABLE 49,50,51,52  /* VARIABLE 53,54,55,56  /* VARIABLE 57,58,59,60  /* VARIABLE 61,62,63,64  /* VARIABLE 65,66,67,68  /* VARIABLE 69,70,71,72  /* VARIABLE 73,74,75,76  /* VARIABLE 77,78,79,80  /* VARIABLE 81,82,83,84  /* VARIABLE 85,86,87,88  /* VARIABLE 89,90,91,92  /* VARIABLE 93,94,95,96  /* VARIABLE 97,98,99,100  /* VARIABLE 101,102,103,104  /* VARIABLE 105,106,107,108  /* VARIABLE 109,110,111,112  /* VARIABLE 113,114,115,116  /* VARIABLE 117,118,119,120  /* VARIABLE 121,122,123,124  /* VARIABLE 125,126,127,128  /* VARIABLE 129,130,131,132  /* VARIABLE 133,134,135,136  /* VARIABLE 137,138,139,140  /* VARIABLE 141,142,143,144  /* VARIABLE 145,146,147,148  /* VARIABLE 149,150,151,152  /* VARIABLE 153,154,155,156  /* VARIABLE 157,158,159,160  /* VARIABLE 161,162,163,164  /* VARIABLE 165,166,167,168  /* VARIABLE 169,170,171,172  /* VARIABLE 173,174,175,176  /* VARIABLE 177,178,179,180  /* VARIABLE 181,182,183,184  /* VARIABLE 185,186,187,188  /* VARIABLE 189,190,191,192  /* VARIABLE 193,194,195,196  /* VARIABLE 197,198,199,200  /* VARIABLE 201,202,203,204  /* VARIABLE 205,206,207,208  /* VARIABLE 209,210,211,212  /* VARIABLE 213,214,215,216  /* VARIABLE 217,218,219,220  /* VARIABLE 221,222,223,224  /* VARIABLE 225,226,227,228  /* VARIABLE 229,230,231,232  /* VARIABLE 233,234,235,236  /* VARIABLE 237,238,239,240  /* VARIABLE 241,242,243,244  /* VARIABLE 245,246,247,248  /* VARIABLE 249,250,251,252  /* VARIABLE 253,254,255,256  /* VARIABLE 257,258,259,260  /* VARIABLE 261,262,263,264  /* VARIABLE 265,266,267,268  /* VARIABLE 269,270,271,272  /* VARIABLE 273,274,275,276  /* VARIABLE 277,278,279,280  /* VARIABLE 281,282,283,284  /* VARIABLE 285,286,287,288  /* VARIABLE 289,290,291,292  /* VARIABLE 293,294,295,296  /* VARIABLE 297,298,299,300  /* VARIABLE 301,302,303,304  /* VARIABLE 305,306,307,308  /* VARIABLE 309,310,311,312  /* VARIABLE 313,314,315,316  /* VARIABLE 317,318,319,320  /* VARIABLE 321,322,323,324  /* VARIABLE 325,326,327,328  /* VARIABLE 329,330,331,332  /* VARIABLE 333,334,335,336  /* VARIABLE 337,338,339,340  /* VARIABLE 341,342,343,344  /* VARIABLE 345,346,347,348  /* VARIABLE 349,350,351,352  /* VARIABLE 353,354,355,356  /* VARIABLE 357,358,359,360  /* VARIABLE 361,362,363,364  /* VARIABLE 365,366,367,368  /* VARIABLE 369,370,371,372  /* VARIABLE 373,374,375,376  /* VARIABLE 377,378,379,380  /* VARIABLE 381,382,383,384  /* VARIABLE 385,386,387,388  /* VARIABLE 389,390,391,392  /* VARIABLE 393,394,395,396  /* VARIABLE 397,398,399,400  /* VARIABLE 401,402,403,404  /* VARIABLE 405,406,407,408  /* VARIABLE 409,410,411,412  /* VARIABLE 413,414,415,416  /* VARIABLE 417,418,419,420  /* VARIABLE 421,422,423,424  /* VARIABLE 425,426,427,428  /* VARIABLE 429,430,431,432  /* VARIABLE 433,434,435,436  /* VARIABLE 437,438,439,440  /* VARIABLE 441,442,443,444  /* VARIABLE 445,446,447,448  /* VARIABLE 449,450,451,452  /* VARIABLE 453,454,455,456  /* VARIABLE 457,458,459,460  /* VARIABLE 461,462,463,464  /* VARIABLE 465,466,467,468  /* VARIABLE 469,470,471,472  /* VARIABLE 473,474,475,476  /* VARIABLE 477,478,479,480  /* VARIABLE 481,482,483,484  /* VARIABLE 485,486,487,488  /* VARIABLE 489,490,491,492  /* VARIABLE 493,494,495,496  /* VARIABLE 497,498,499,500  /* VARIABLE 501,502,503,504  /* VARIABLE 505,506,507,508  /* VARIABLE 509,510,511,512  /* VARIABLE 513,514,515,516  /* VARIABLE 517,518,519,520  /* VARIABLE 521,522,523,524  /* VARIABLE 525,526,527,528  /* VARIABLE 529,530,531,532  /* VARIABLE 533,534,535,536  /* VARIABLE 537,538,539,540  /* VARIABLE 541,542,543,544  /* VARIABLE 545,546,547,548  /* VARIABLE 549,550,551,552  /* VARIABLE 553,554,555,556  /* VARIABLE 557,558,559,560  /* VARIABLE 561,562,563,564  /* VARIABLE 565,566,567,568  /* VARIABLE 569,570,571,572  /* VARIABLE 573,574,575,576  /* VARIABLE 577,578,579,580  /* VARIABLE 581,582,583,584  /* VARIABLE 585,586,587,588  /* VARIABLE 589,590,591,592  /* VARIABLE 593,594,595,596  /* VARIABLE 597,598,599,600  /* VARIABLE 601,602,603,604  /* VARIABLE 605,606,607,608  /* VARIABLE 609,610,611,612  /* VARIABLE 613,614,615,616  /* VARIABLE 617,618,619,620  /* VARIABLE 621,622,623,624  /* VARIABLE 625,626,627,628  /* VARIABLE 629,630,631,632  /* VARIABLE 633,634,635,636  /* VARIABLE 637,638,639,640  /* VARIABLE 641,642,643,644  /* VARIABLE 645,646,647,648  /* VARIABLE 649,650,651,652  /* VARIABLE 653,654,655,656  /* VARIABLE 657,658,659,660  /* VARIABLE 661,662,663,664  /* VARIABLE 665,666,667,668  /* VARIABLE 669,670,671,672  /* VARIABLE 673,674,675,676  /* VARIABLE 677,678,679,680  /* VARIABLE 681,682,683,684  /* VARIABLE 685,686,687,688  /* VARIABLE 689,690,691,692  /* VARIABLE 693,694,695,696  /* VARIABLE 697,698,699,700  /* VARIABLE 701,702,703,704  /* VARIABLE 705,706,707,708  /* VARIABLE 709,710,711,712  /* VARIABLE 713,714,715,716  /* VARIABLE 717,718,719,720  /* VARIABLE 721,722,723,724  /* VARIABLE 725,726,727,728  /* VARIABLE 729,730,731,732  /* VARIABLE 733,734,735,736  /* VARIABLE 737,738,739,740  /* VARIABLE 741,742,743,744  /* VARIABLE 745,746,747,748  /* VARIABLE 749,750,751,752  /* VARIABLE 753,754,755,756  /* VARIABLE 757,758,759,760  /* VARIABLE 761,762,763,764  /* VARIABLE 765,766,767,768  /* VARIABLE 769,770,771,772  /* VARIABLE 773,774,775,776  /* VARIABLE 777,778,779,780  /* VARIABLE 781,782,783,784  /* VARIABLE 785,786,787,788  /* VARIABLE 789,790,791,792  /* VARIABLE 793,794,795,796  /* VARIABLE 797,798,799,800  /* VARIABLE 801,802,803,804  /* VARIABLE 805,806,807,808  /* VARIABLE 809,810,811,812  /* VARIABLE 813,814,815,816  /* VARIABLE 817,818,819,820  /* VARIABLE 821,822,823,824  /* VARIABLE 825,826,827,828  /* VARIABLE 829,830,831,832  /* VARIABLE 833,834,835,836  /* VARIABLE 837,838,839,840  /* VARIABLE 841,842,843,844  /* VARIABLE 845,846,847,848  /* VARIABLE 849,850,851,852  /* VARIABLE 853,854,855,856  /* VARIABLE 857,858,859,860  /* VARIABLE 861,862,863,864  /* VARIABLE 865,866,867,868  /* VARIABLE 869,870,871,872  /* VARIABLE 873,874,875,876  /* VARIABLE 877,878,879,880  /* VARIABLE 881,882,883,884  /* VARIABLE 885,886,887,888  /* VARIABLE 889,890,891,892  /* VARIABLE 893,894,895,896  /* VARIABLE 897,898,899,900  /* VARIABLE 901,902,903,904  /* VARIABLE 905,906,907,908  /* VARIABLE 909,910,911,912  /* VARIABLE 913,914,915,916  /* VARIABLE 917,918,919,920  /* VARIABLE 921,922,923,924  /* VARIABLE 925,926,927,928  /* VARIABLE 929,930,931,932  /* VARIABLE 933,934,935,936  /* VARIABLE 937,938,939,940  /* VARIABLE 941,942,943,944  /* VARIABLE 945,946,947,948  /* VARIABLE 949,950,951,952  /* VARIABLE 953,954,955,956  /* VARIABLE 957,958,959,960  /* VARIABLE 961,962,963,964  /* VARIABLE 965,966,967,968  /* VARIABLE 969,970,971,972  /* VARIABLE 973,974,975,976  /* VARIABLE 977,978,979,980  /* VARIABLE 981,982,983,984  /* VARIABLE 985,986,987,988  /* VARIABLE 989,990,991,992  /* VARIABLE 993,994,995,996  /* VARIABLE 997,998,999,1000
```

C* for Multiple-Value Fields Within Periodic Groups

For a multiple-value field within a periodic group, you can also define a C* variable with an index range specification.

The following examples use the multiple-value field BONUS, which is part of the periodic group INCOME. All three examples yield the same result.

Example 1 - Reporting Mode:

```
READ EMPLOYEES BY PERSONNEL-ID FROM 11100117     OBTAIN C*BONUS (1:3)           BONUS (1:3,1:3) *     DISPLAY C*BONUS (1:3)            BONUS (1:3,1:3) END
```

 Copyright © Software AG 2002

Example 2 - Structured Mode:

```
DEFINE DATA LOCAL 1 EMP VIEW OF EMPLOYEES   2 PERSONNEL-ID   2 INCOME (1:3)     3 C*BONUS     3 BONUS (1:3) END-DEFINE READ EMP BY PERSONNEL-ID FROM 11100117    DISPLAY C*BONUS (1:3)           BONUS (1:3,1:3) END-READ END
```

Example 3 - Structured Mode:

```
DEFINE DATA LOCAL 1 EMP VIEW OF EMPLOYEES   2 PERSONNEL-ID   2 C*BONUS (1:3)   2 INCOME (1:3)     3 BONUS (1:3) END-DEFINE READ EMP BY PERSONNEL-ID FROM 11100117    DISPLAY PERSONNEL-ID C*BONUS (*) BONUS (*,*) END-READ END
```

Note:
As the Adabas format buffer does not permit ranges for count fields, they are generated as individual fields; therefore a C* index range for a large array may cause an Adabas format buffer overflow.

Qualifying Data Structures

To identify a field when referencing it, you may qualify the field; that is, before the field name, you specify the name of the level-1 data element in which the field is located and a period.

If a field cannot be identified uniquely by its name (for example, if the same field name is used in multiple groups/views), you must qualify the field when you reference it.

The combination of level-1 data element and field name must be unique.

Example:

```
DEFINE DATA LOCAL 1 FULL-NAME   2 LAST-NAME (A20)   2 FIRST-NAME (A15) 1 OUTPUT-NAME   2 LAST-NAME (A20)   2 FIRST-NAME (A15) END-DEFINE ... MOVE FULL-NAME.LAST-NAME TO OUTPUT-NAME.LAST-NAME ...
```

The qualifier must be a level-1 data element.

Example:

```
DEFINE DATA LOCAL 1 GROUP1   2 SUB-GROUP     3 FIELD1 (A15)     3 FIELD2 (A15) END-DEFINE ... MOVE 'ABC' TO GROUP1.FIELD1 ...
```

Note:
If you use the same name for a user-defined variable and a database field (which you should not do anyway), you must qualify the database field when you want to reference it; because if you do not, the user-defined variable will be referenced instead.

Constants

- Numeric Constants
- Alphanumeric Constants
- Date and Time Constants
- Hexadecimal Constants
- Logical Constants
- Floating Point Constants
- Handle Constants

Constants are used throughout Natural programs. This section discusses the types of constants that are supported and how they are used.

Numeric Constants

A numeric constant may contain 1 to 29 numeric digits. A numeric constant used in a COMPUTE, MOVE, or arithmetic statement may contain a decimal point and sign notation.

Examples:

```
MOVE 3 TO #XYZ COMPUTE #PRICE = 23.34 COMPUTE #XYZ = -103 COMPUTE #A = #B * 6074
```

Note:
Internally, numeric constants without decimal digits are represented in integer form (format I), while numeric constants with decimal digits, as well as numeric constants without decimal digits that are too large to fit into format I, are represented in packed form (format P).
On mainframe computers, numeric constants are represented internally in packed form (format P); exception: if a numeric constant is used in an arithmetic operation in which the other operand is an integer variable (format I), the numeric constant is represented in integer form (format I).

Validation of Numeric Constants

When numeric constants are used within one of the statements MOVE, COMPUTE, or DEFINE DATA with INIT option, Natural checks at compilation time whether a constant value fits into the corresponding field. This avoids runtime errors in situations where such an error condition can already be detected during compilation.

Alphanumeric Constants

An alphanumeric constant may contain 1 to 253 alphanumeric characters.

An alphanumeric constant must be enclosed in either apostrophes (’) or quotation marks (").

Examples:

```
MOVE ’ABC’ TO #FIELDX MOVE ’% INCREASE’ TO #TITLE DISPLAY "LAST-NAME" NAME
```

An alphanumeric constant that is used to assign a value to a user-defined variable must not be split between statement lines.

Apostrophes Within Alphanumeric Constants

If you want an apostrophe to be part of an alphanumeric constant that is enclosed in apostrophes, you must write this as two apostrophes or as a single quotation mark.
If you want an apostrophe to be part of an alphanumeric constant that is enclosed in quotation marks, you write this as a single apostrophe.

Example:

If you want the following to be output:

```
HE
SAID, ’HELLO’
```

you can use any of the following notations:

```
WRITE ’HE SAID, ’’HELLO’’’ WRITE ’HE SAID, "HELLO"’ WRITE "HE SAID,
""HELLO""" WRITE "HE SAID, ’HELLO’"
```

 Copyright © Software AG 2002

Note:
If quotation marks are not converted to apostrophes as shown above, this is due to the setting of the profile parameter TQ; ask your Natural administrator for details.

Concatenation of Alphanumeric Constants

Alphanumeric constants may be concatenated to form a single value by use of a hyphen.

Examples:

```
MOVE 'XXXXXX' -      'YYYYYY' TO #FIELD  MOVE "ABC" - 'DEF' TO #FIELD
```

In this way, alphanumeric constants can also be concatenated with hexadecimal constants.

Date and Time Constants

A date constant may be used in conjunction with a format D variable. Date constants may have the following formats:

D'YYYY-MM-DD'	International date format
D'DD.MM.YYYY'	German date format
D'DD/MM/YYYY'	European date format
D'MM/DD/YYYY'	USA date format

Example:

```
DEFINE DATA LOCAL 1 #DATE (D) END-DEFINE ... MOVE D'1997-04-27' TO #DATE ...
```

The default date format is controlled by the profile parameter DTFORM as set by the Natural administrator.

A time constant may be used in conjunction with a format T variable. A time constant has the following format:

T'*hh:ii:ss*'

where

Character	Explanation
hh	hours
ii	minutes
ss	seconds

Example:

```
DEFINE DATA LOCAL 1 #TIME (T) END-DEFINE ... MOVE T'11:33:00' TO #TIME ...
```

Extended Time Constants

A time variable (format T) can contain date and time information, date information being a subset of time information; however, with a "normal" time constant (prefix "T") only the time information of a time variable can be handled:

T'*hh:ii:ss*'

With an extended time constant (prefix "E"), it is possible to handle the full content of a time variable, including the date information:

E'*yyyy-mm-dd hh:ii:ss*'

Apart from that, the use of an extended time constant in conjunction with a time variable is the same as for a normal time constant.

Note:
The format in which the date information has to be specified in an extended time constant depends on the setting of the profile parameter DTFORM. The extended time constant shown above assumes DTFORM=I (international date format).

Hexadecimal Constants

A hexadecimal constant may be used to enter a value which cannot be entered as a standard keyboard character.

A hexadecimal constant is prefixed with an "H". The constant itself must be enclosed in apostrophes and may consist of the hexadecimal characters 0 - 9, A - F. Two hexadecimal characters are required to represent one byte of data.

The hexadecimal representation of a character varies depending on whether your computer uses an ASCII or EBCDIC character set. When you transfer hexadecimal constants to another computer, you may therefore have to convert the characters.

ASCII Examples:

```
H'313233'    (equivalent to the alphanumeric constant '123') H'414243'    (equivalent to the alphanumeric constant 'ABC')
```

EBCDIC Examples:

```
H'F1F2F3'    (equivalent to the alphanumeric constant '123') H'C1C2C3'    (equivalent to the alphanumeric constant 'ABC')
```

Hexadecimal constants may be concatenated by using a hyphen between the constants.

ASCII Examples:

```
H'414243' - H'444546' (equivalent to 'ABCDEF')
```

EBCDIC Examples:

```
H'C1C2C3' - H'C4C5C6' (equivalent to 'ABCDEF')
```

In this way, hexadecimal constants can also be concatenated with alphanumeric constants.

Note:
When a hexadecimal constant is transferred to another field, it will be treated as an alphanumeric value.
Under UNIX, if a hexadecimal constant is output that contains any characters from the ranges H'00' to H'1F' or H'80' to H'A0', these characters will not be output, as they would be interpreted as terminal control characters. As of version 2.2 these hex constants are not suppressed.

 Copyright © Software AG 2002

Logical Constants

The logical constants "TRUE" and "FALSE" may be used to assign a logical value to a variable defined with format L.

Example:

```
DEFINE DATA LOCAL 1 #FLAG (L) END-DEFINE ... MOVE TRUE TO #FLAG ... IF #FLAG ...    statement ...    MOVE FALSE TO #FLAG END-IF ...
```

Floating Point Constants

Floating point constants can be used with variables defined with format F.

Example:

```
DEFINE DATA LOCAL 1 #FLT1 (F4) END-DEFINE ... COMPUTE #FLT1 = -5.34E+2 ...
```

See information on arithmetic involving floating-point numbers.

Attribute Constants

Attribute constants can be used with variables defined with C format. This type of constant must be enclosed within parentheses.

The following attributes may be used:

AD=D	default	**CD=BL**	blue
AD=B	blinking	**CD=GR**	green
AD=I	intensified	**CD=NE**	neutral
AD=N	non-display	**CD=PI**	pink
AD=V	reverse video	**CD=RE**	red
AD=U	underlined	**CD=TU**	turquoise
AD=C	cursive/italic	**CD=YE**	yellow
AD=Y	dynamic attribute		
AD=P	protected		

Example:

```
DEFINE DATA LOCAL 1 #ATTR (C) 1 #FIELD (A10) END-DEFINE  ... MOVE (AD=I CD=BL) TO #ATTR ... INPUT #FIELD (CV=#ATTR) ...
```

Handle Constants

The handle constant NULL-HANDLE can be used with GUI handles and object handles.

For further information on GUI handles, see the Natural User’s Guide for Windows. For further information on object handles, see the NaturalX documentation.

Report Specification - *rep*

"*(rep)*" is the output report identifier for which a statement is applicable. If a Natural program is to produce multiple reports, the notation "*(rep)*" must be specified with each output statement which is to be used to create output for any report other than the first report (report 0). A value of 1 - 31 may be specified.

On mainframe computers, this notation only applies to reports created in batch mode, to reports under Com-plete, CMS, IMS/TM or TIAM; or when using Natural Advanced Facilities under CICS, TSO or UTM.

Examples:

```
DISPLAY (1) NAME ...WRITE (4) NAME ...
```

The value for (rep) may also be a logical name which has been assigned using the DEFINE PRINTER statement.

Example:

```
DEFINE PRINTER (LIST=5) OUTPUT 'LPT1'WRITE (LIST) NAME ...
```

Text Notation

'text' specifies text to be used in conjunction with an INPUT, DISPLAY, WRITE, WRITE TITLE or WRITE TRAILER statement. The text must be enclosed in either apostrophes (') or quotation marks ("). The text itself may be 1 to 72 characters and must not be continued from one line to the next. Text elements may be concatenated by using a hyphen.

Examples:

```
REINPUT 'PLEASE ENTER A VALID VALUE' WRITE    "NEW SALARY" #NEW-SALARY WRITE    'TEXT1'-'TEXT2'-'TEXT3'
```

If you want an apostrophe to be part of a text string that is enclosed in apostrophes, you must write this as two apostrophes or as a single quotation mark. Either notation will be output as a single apostrophe.
If you want an apostrophe to be part of a text string that is enclosed in quotation marks, you write this as a single apostrophe.

Examples:

```
#FIELDA = 'O''CONNOR' #FIELDA = 'O"CONNOR' #FIELDA = "O'CONNOR"
```

In all three cases, the result will be:

```
O'CONNOR
```

Note:
If quotation marks are not converted to apostrophes as shown above, this is due to the setting of the profile parameter TQ; ask your Natural administrator for details.

If a single character is to be output several times as text, you use the following notation:

`'c'(n)`

As c you specify the character, and as n the number of times the character is to be generated. The maximum value for n is 249.

 Copyright © Software AG 2002

Example:

```
WRITE '*'(3)
```

Instead of apostrophes before and after the character c you can also use quotation marks.

User Comments

You have the following possibilities for entering your comments in source code:

- If you wish to use an entire source-code line for a user comment, you enter one of the following at the beginning of the line:
  - an asterisk and a blank (*),
  - two asterisks (**), or
  - a slash and an asterisk (/*):
    ```
    *  USER COMMENT
    ** USER COMMENT
    /* USER COMMENT
    ```
- If you wish to use only the latter part of a source-code line for a user comment, you enter a blank, a slash and an asterisk (/*); the remainder of the line after this notation is thus marked as a comment:
  ```
  ADD 5 TO #A          /* USER COMMENT
  ```

End of a Statement

To explicitly mark the end of a statement, you can place a semicolon (;) between the statement and the next statement. This can be used to make the program structure clearer, but is not required.

Logical Condition Criteria

- Relational Expression
- Extended Relational Expression
- MASK Option
- SCAN Option
- BREAK Option
- IS Option
- Evaluation of a Logical Variable
- Modified Control Variables
- SPECIFIED Option
- Fields Used Within Logical Condition Criteria
- Logical Operators in Complex Logical Expressions

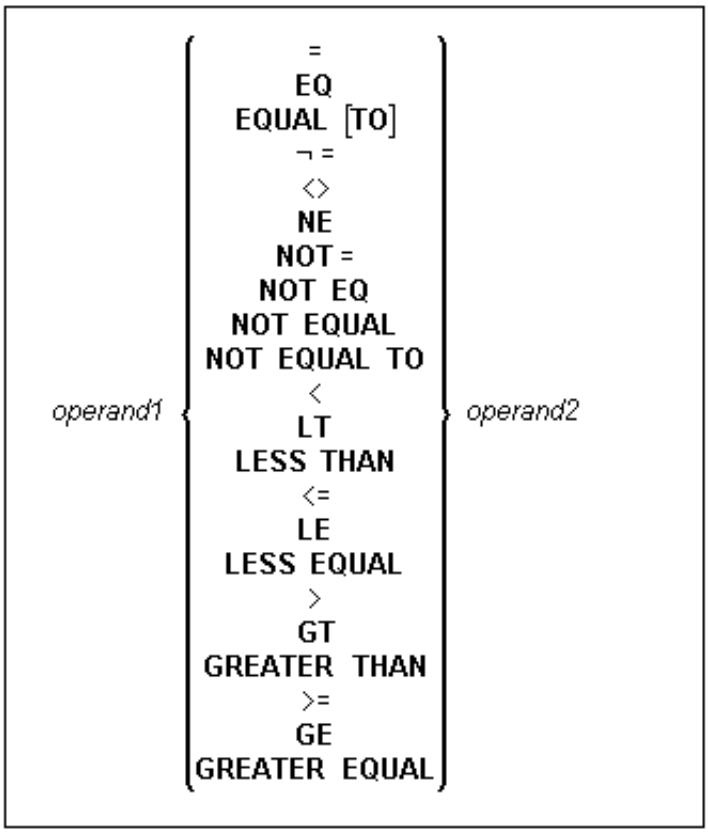
The basic criterion is a relational expression. Multiple relational expressions may be combined with logical operators (AND, OR) to form complex criteria.

Arithmetic expressions may also be used to form a relational expression.

Logical condition criteria can be used in the following statements:

Statement	Usage
FIND	A WHERE clause containing logical condition criteria may be used to indicate criteria in addition to the basic selection criteria as specified in the WITH clause. The logical condition criteria specified with the WHERE clause are evaluated after the record has been selected and read. In a WITH clause, "basic search criteria" (as described with the FIND statement) are used, but not logical condition criteria.
READ	A WHERE clause containing logical condition criteria may be used to specify whether a record that has just been read is to be processed. The logical condition criteria are evaluated after the record has been read.
HISTOGRAM	A WHERE clause containing logical condition criteria may be used to specify whether the value that has just been read is to be processed. The logical condition criteria are evaluated after the value has been read.
ACCEPT/REJECT	An IF clause may be used with an ACCEPT or REJECT statement to specify logical condition criteria in addition to that specified when the record was selected/read with a FIND, READ, or HISTOGRAM statement. The logical condition criteria are evaluated after the record has been read and after record processing has started.
IF	Logical condition criteria are used to control statement execution.
DECIDE FOR	Logical condition criteria are used to control statement execution.
REPEAT	The UNTIL or WHILE clause of a REPEAT statement contain logical condition criteria which determine when a processing loop is to be terminated.

Relational Expression



 Copyright © Software AG 2002

Operand	Possible Structure						Possible Formats												Referencing Permitted	Dynamic Definition
Operand1	C	S	A		N	E	A	N	P	I	F	B	D	T	L		G	O	yes	yes
Operand2	C	S	A		N	E	A	N	P	I	F	B	D	T	L		G	O	yes	no

For an explanation of the Operand Definition Table shown above, see Syntax Symbols and Operand Definition Tables in the Natural Statements documentation. In the "Possible Structure" section of the table above, "E" stands for arithmetic expressions; that is, any arithmetic expression may be specified as an operand within the relational expression.

Examples:

```
IF NAME = 'SMITH' IF LEAVE-DUE GT 40 IF NAME = #NAME
```

For information on comparing arrays in a relational expression, see Processing of Arrays.

Note:
If a floating-point operand is used, comparison is performed in floating point. Floating-point numbers as such have only a limited precision; therefore, rounding/truncation errors cannot be precluded when numbers are converted to/from floating-point representation.

Arithmetic Expressions in Logical Conditions

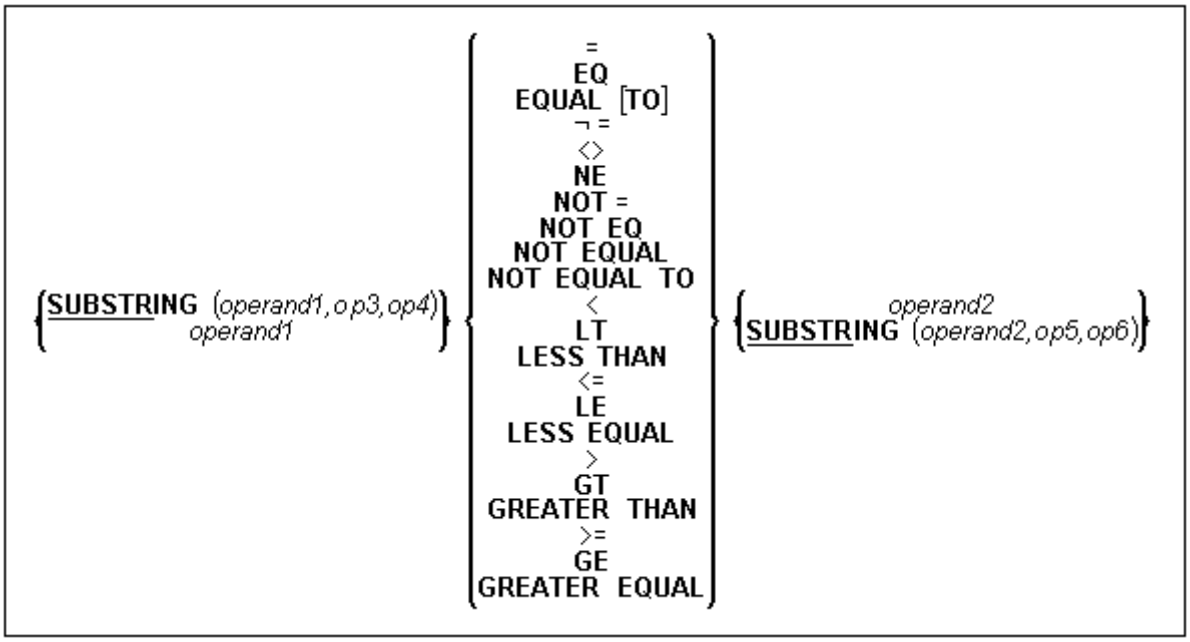
The following example shows how arithmetic expressions can be used in logical conditions:

```
IF #A + 3 GT #B - 5 AND #C * 3 LE #A + #B
```

Handles in Logical Conditions

If the operands in a relation expression are handles, only EQUAL and NOT EQUAL operators may be used.

SUBSTRING Option in Relational Expression



Operand	Possible Structure					Possible Formats											Referencing Permitted	Dynamic Definition
Operand1	C	S	A		N	A											yes	yes
Operand2	C	S	A		N	A											yes	no
Op3/Op5	C	S					N	P	I								yes	no
Op4/Op6	C	S					N	P	I								yes	no

With the SUBSTRING option, you can compare a *part* of an alphanumeric field. After the field name *(operand1)* you specify first the starting position *(op3)* and then the length *(op4)* of the field portion to be compared.

Also, you can compare a field value with part of another field value. After the field name *(operand2)* you specify first the starting position *(op5)* and then the length *(op6)* of the field portion *operand1* is to be compared with.

You can also combine both forms, that is, you can specify a SUBSTRING for both *operand1* and *operand2*.

Examples:

This expression compares the 5th to 12th position inclusive of the value in field #A with the value of field #B:

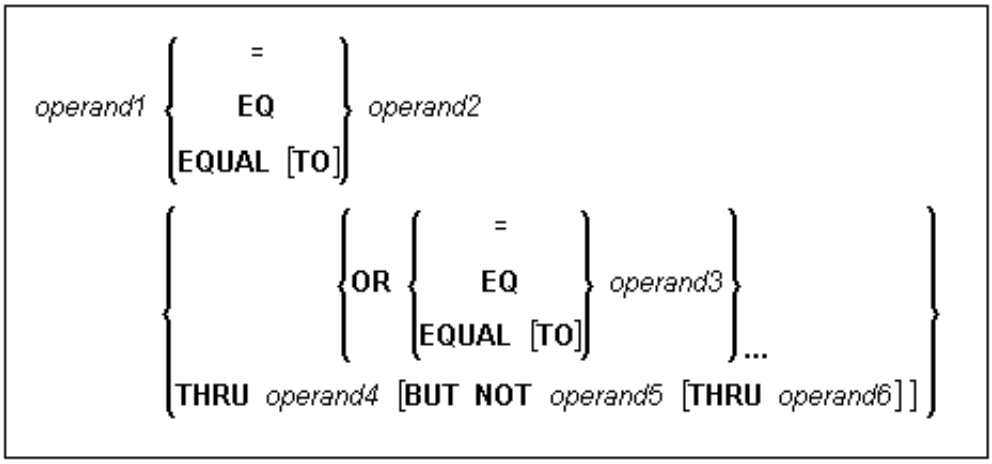
```
SUBSTRING(#A,5,8) = #B
```

This expression compares the value of field #A with the 3rd to 6th position inclusive of the value in field #B:

```
#A = SUBSTRING(#B,3,4)
```

Note:
If you omit op3/op5, the starting position is assumed to be "1". If you omit op4/op6, the length is assumed to be from the starting position to the end of the field.

Extended Relational Expression



 Copyright © Software AG 2002

Operand	Possible Structure					Possible Formats												Referencing Permitted	Dynamic Definition
Operand1		S	A		N* E	A	N	P	I	F	B	D	T			G	O	yes	no
Operand2	C	S	A		N* E	A	N	P	I	F	B	D	T			G	O	yes	no
Operand3	C	S	A		N* E	A	N	P	I	F	B	D	T			G	O	yes	no
Operand4	C	S	A		N* E	A	N	P	I	F	B	D	T			G	O	yes	no
Operand5	C	S	A		N* E	A	N	P	I	F	B	D	T			G	O	yes	no
Operand6	C	S	A		N* E	A	N	P	I	F	B	D	T			G	O	yes	no

* Mathematical functions and system variables are permitted.
Break functions are not permitted.

Operand3 can also be specified using a MASK or SCAN option; that is, it can be specified as:

MASK *(mask-definition) [operand]*
MASK *operand*
SCAN *operand*

For details on these options, see the sections MASK Option and SCAN Option.

Examples:

```
IF #A = 2 OR = 4 OR = 7 IF #A = 5 THRU 11 BUT NOT 7 THRU 8
```

MASK Option

With the MASK option, you can check selected positions of a field for a specific content.

Constant Mask

operand1 { = | EQ | EQUAL TO | NE | NOT EQUAL } **MASK** *(mask-definition)* [*operand 2*]

Operand	Possible Structure					Possible Formats												Referencing Permitted	Dynamic Definition
Operand1		S	A		N	A	N	P										yes	no
Operand2	C	S				A	N	P			B							yes	no

Operand2 can only be used if the *mask-definition* contains at least one "X". *Operand1* and *operand2* must be format-compatible: if *operand1* is of format A, *operand2* must be of format A, B or N; if *operand1* is of format N or P, *operand2* must be of format N or P. An "X" in the the *mask-definition* selects the corresponding positions of the content of *operand1* and *operand2* for comparison.

Variable Mask

Apart from a constant *mask-definition* (see above), you may also specify a variable mask definition:

```
                  =
                 EQ
operand1  EQUAL  TO  MASK  operand2
                 NE
            NOT EQUAL
```

Operand	Possible Structure					Possible Formats											Referencing Permitted	Dynamic Definition
Operand1		S	A		N	A	N	P									yes	no
Operand2		S				A											yes	no

The content of *operand2* will be taken as the mask definition. Trailing blanks in *operand2* will be ignored.

Characters in a Mask

The following characters may be used within a mask definition (the mask definition is contained in *mask-definition* for a constant mask and *operand2* for a variable mask):

 Copyright © Software AG 2002

Character	Meaning
. or ? or _	Indicates a single position that is not to be checked.
* or %	Indicates any number of positions not to be checked.
/	(Slash) Used to check if a value ends with a specific character (or string of characters). For example, the following condition will be true if there is either an "E" in the last position of the field, or the last "E" in the field is followed by nothing but blanks: IF #FIELD = MASK (*'E'/)
A	The position is to be checked for an alphabetical character (upper or lower case).
'*c*'	One or more positions are to be checked for the characters bounded by apostrophes (a double apostrophe indicates that a single apostrophe is the character to be checked for).
C	The position is to be checked for an alphabetical character (upper or lower case), a numeric character, or a blank.
DD	The two positions are to be checked for a valid day notation (01 - 31; dependent on the values of MM and YY/YYYY, if specified; see also Checking Dates).
H	The position is to be checked for hexadecimal content (A - F, 0 - 9).
L	The position is to be checked for a lower-case alphabetical character (a - z).
MM	The positions are to be checked for a valid month (01 - 12).
N	The position is to be checked for a numeric digit.
n...	One (or more) positions are to be checked for a numeric value in the range 0 - n.
n1-n2 or *n1:n2*	The positions are checked for a numeric value in the range n1-n2. n1 and n2 must be of the same length.
P	The position is to be checked for a displayable character (U, L, N or S).
S	The position is to be checked for special characters.
U	The position is to be checked for an upper-case alphabetical character (A - Z).
X	The position is to be checked against the equivalent position in the value (operand2) following the mask-definition. "X" is not allowed in a variable mask definition, as it makes no sense.
YY	The two positions are to be checked for a valid year (00 - 99). See also Checking Dates.
YYYY	The four positions are checked for a valid year (0000 - 2699).
Z	The position is to be checked for a character whose left half-byte is hexadecimally 3 or 7 (ASCII) or A - F (EBCDIC), and whose right half-byte is hexadecimally 0 - 9. This may be used to correctly check for numeric digits in negative numbers. With "N" (which indicates a position to be checked for a numeric digit), a check for numeric digits in negative numbers leads to incorrect results, because the sign of the number is stored in the last digit of the number, causing that digit to be hexadecimally represented as non-numeric. Within a mask, use only one "Z" for each sequence of numeric digits that is checked.

Mask Length

The length of the mask determines how many positions are to be checked.

Example:

```
DEFINE DATA LOCAL 1 #CODE (A15) END-DEFINE ... IF #CODE = MASK (NN'ABC'....NN) ...
```

The first two positions of #CODE are to be checked for numeric content. The three following positions are checked for the contents "ABC". The next four positions are not to be checked. Positions ten and eleven are to be checked for numeric content. Positions twelve to fifteen are not to be checked.

Checking Dates

Only one date may be checked within a given mask.

When dates are checked for a day (DD) and no month (MM) is specified in the mask, the current month will be assumed.

When dates are checked for a day (DD) and no year (YY or YYYY) is specified in the mask, the current year will be assumed.

When dates are checked for a 2-digit year (YY), the current century will be assumed if the profile parameter YSLW is set to "0". If the YSLW parameter is set to another value, the century will be determined by the "year sliding window" (as described under profile parameter YSLW in your Natural Operations documentation).

Examples:

Example 1:

```
MOVE 1131 TO #DATE (N4) IF #DATE = MASK (MMDD)
```

In this example, month and day are checked for validity. The value for month (11) will be considered valid, whereas the value for day (31) will be invalid since the 11th month has only 30 days.

Example 2:

```
IF #DATE(A8) = MASK (MM'/'DD'/'YY)
```

In this example, the content of the field #DATE is be checked for a valid date with the format MM/DD/YY (month/day/year).

Example 3:

```
IF #DATE (A4) = MASK (19-20YY)
```

In this example, the content of field #DATE is checked for a two-digit number in the range 19 to 20 followed by a valid two-digit year (00 through 99). The century is supplied by Natural as described above.
Note: Although apparent, the above mask does not allow to check for a valid year in the range 1900 through 2099, because the numeric value range 19-20 is checked independent of the year validation.
To check for year ranges, code one check for the date validation and another for the range validation:

```
IF #DATE (A10) = MASK (YYYYŸ-‡MMŸ-‡DD) AND #DATE = MASK (19-20)
```

 Copyright © Software AG 2002

Checking Against the Content of Constants or Variables

If the value for the mask check is to be taken from either a constant or a variable, this value *(operand2)* must be specified immediately following the *mask-definition*.

Operand2 must be at least as long as the mask.

In the mask, you indicate each position to be checked with "X", and each position not to be checked with "." (or "?" or "_").

Example:

```
DEFINE DATA LOCAL 1 #NAME (A15) END-DEFINE ... IF #NAME = MASK (..XX) 'ABCD' ...
```

It is checked whether the field #NAME contains "CD" in the third and fourth positions. Positions one and two are not checked.

The length of the mask determines how many positions are to be checked. The mask is left-justified against any field or constant used in the mask operation. The format of the field (or constant) on the right side of the expression must be the same as the format of the field on the left side of the expression.

If the field to be checked (operand1) is of format A, any constant used (operand2) must be enclosed in apostrophes. If the field is numeric, the value used must be a numeric constant or the content of a numeric database field or user-defined variable.

In either case, any characters/digits within the value specified which do not match positionally the "X" indicator within the mask are ignored.

The result of the MASK operation is true when the indicated positions in both values are identical.

Example:

```
/* EXAMPLE 'LICMASK' /* EXAMPLE OF USING MASK OPTION WITHIN LOGICAL CONDITION /************************************************** DEFINE DATA LOCAL 1 EMPLOY-VIEW VIEW OF EMPLOYEES  2 CITY END-DEFINE /************************************************** HISTOGRAM EMPLOY-VIEW CITY  IF CITY = MASK (....XX) '....NN'  DISPLAY NOTITLE CITY 'NUMBER  END-IF END-HISTOGRAM /************************************************** END
```

In the above example, the record will be accepted if the fifth and sixth positions of the field CITY each contain the character "N".

Range Checks

When performing range checks, the number of positions verified in the supplied variable is defined by the precision of the value supplied in the mask specification. For example, a mask of (...193...) will verify positions 4 to 6 for a three-digit number in the range 000 to 193.

Additional Examples of Mask Definitions:

- In this example, each character of #NAME is checked for an alphabetical character:
  ```
  IF #NAME (A10) = MASK (AAAAAAAAAA)
  ```
- In this example, positions 4 to 6 of #NUMBER are checked for a numeric value:
  ```
  IF #NUMBER (A6) = MASK (...NNN)
  ```
- In this example, positions 4 to 6 of #VALUE are to be checked for the value "123":
  ```
  IF #VALUE(A10) = MASK (...'123')
  ```
- This example will check if #LICENSE contains a license number which begins with "NY-" and whose last five characters are identical to the last five positions of #VALUE:
  ```
  DEFINE DATA LOCAL
     1 #VALUE(A8)
     1 #LICENSE(A8)
  END-DEFINE
  INPUT 'ENTER KNOWN POSITIONS OF LICENSE PLATE:' #VALUE
  ```

```
IF #LICENSE = MASK ('NY-'XXXXX) #VALUE
```

- The following condition would be met by any value which contains "NAT" and "AL" no matter which and how many other characters are between "NAT" and "AL" (this would include the values Natural and NATIONALITY as well as NATAL):
  `MASK('NAT'*'AL')`

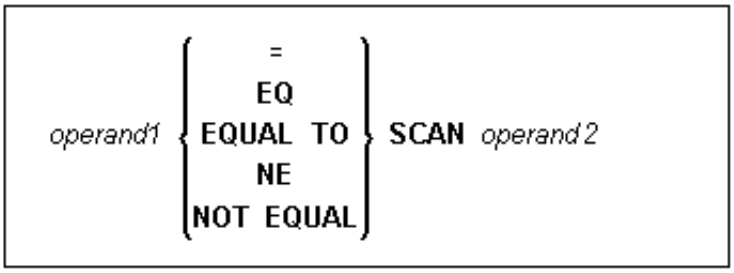
Checking Packed or Unpacked Numeric Data

Legacy applications often have packed or unpacked numeric variables redefined with alphanumeric or binary fields. Such redefinitions are not recommended, because using the packed or unpacked variable in an assignment or computation may lead to errors or unpredictable results. To validate the contents of such a redefined variable before the variable is used, use the N option as many as number of digits - 1 times followed by a single Z option.

Examples:

```
IF #P1 (P1) = MASK (Z)IF #N4 (N4) = MASK (NNNZ)IF #P5 (P5) = MASK (NNNNZ)
```

SCAN Option



Operand	Possible Structure					Possible Formats											Referencing Permitted	Dynamic Definition
Operand1		S	A		N	A	N	P									yes	no
Operand2	C	S				A					B*						yes	no

* *Operand2* may only be binary if *operand1* is alphanumeric.

The SCAN option is used to scan for a specific value within a field.

The characters used in the SCAN option *(operand2)* may be specified as an alphanumeric constant (a character string bounded by apostrophes) or the contents of an alphanumeric database field or user-defined variable.

Trailing blanks are automatically eliminated from the value. Therefore, the SCAN option cannot be used to scan for blanks.

If *operand1* is alphanumeric, *operand2* may also be binary.

The field to be scanned *(operand1)* may be of format A, N or P. The SCAN operation may be specified with the equal (EQ) or not equal (NE) operators.

The length of the character string for the SCAN operation should be less than the length of the field to be scanned. If the length of the character string specified is identical to the length of the field to be scanned, then an EQUAL operator should be used instead of SCAN.

 Copyright © Software AG 2002

Example of SC AS Option:

```
/* EXAMPLE 'LCSCAN' /* EXAMPLE OF USING SCAN OPTION IN LOGICAL CONDITION /*********************************** DEFINE DATA LOCAL 1 EMPLOY-VIEW VIEW OF EMPLOYEES  2 NAME 1 #VALUE (A4) 1 #COMMENT (A40) INIT <' '> END-DEFINE /*********************************** INPUT 'ENTER SCAN VALUE:' #VALUE LIMIT 10 HISTOGRAM EMPLOY-VIEW NAME  RESET #COMMENT  IF NAME = SCAN #VALUE  MOVE 'MATCH' TO #COMMENT  END-IF  DISPLAY NOTITLE NAME *NUMBER #COMMENT END-HISTOGRAM /*********************************** END
ENTER SCAN VALUE: 11

 NAME        NUMBER  #COMMENT                                  ABBILLAN          1  1 MATCHHISTOGRAM          SABOR                  SAMTHOMSON             BASSERBLE              DATHAMERIEU            DAML                   LAMPRECE               LALONDE                LALOTITA               LAXSCHAMER             1
 ---------- ------- ---------------------------------------- ---------- -------- -------- ------------------------ ---------- ---------- ----------
```

This format for which the check is performed can be:

N*ll.ll*	Numeric with length *ll.ll.*
F*ll*	Floating point with length *ll.*
D	Date. The following date formats are possible: *dd-mm-yy*, *dd-mm-yyyy*, *ddmmyyyy* (*dd* = day, *mm* = month, *yy* or *yyyy* = year). The sequence of the day, month and year components as well as the characters between the components are determined by the profile parameter DTFORM (which is described in your Natural Operations documentation).
T	Time (according to the default time display format).
P*ll.ll*	Packed numeric with length *ll.ll*.
I*ll*	Integer with length *ll*.

When the check is performed, leading and trailing blanks in operand1 will be ignored.

The IS option may, for example, be used to check the content of a field before the mathematical function VAL (extract numeric value from an alphanumeric field) is used to ensure that it will not result in a runtime error.

Note:
The IS option cannot be used to check if the value of an alphanumeric field is in the specified "format", but if it can be **converted** to that "format". To check if a value is in a specific format, you can use the MASK option.

```
Example of the Option:
/* EXAMPLE "IS(N8)" /* EXAMPLE OF NUMERICVARIABLE CHECK IN NUMERICDIRECTIONS ------------------------------------------------------------------------------
DEFINE DATA LOCAL 1 #TEXTVAL (A20)                  /* ONLY CHECK IF VAL FUNCTION 1 #TEXTVAL (A20)          /* REGARDING CHECK OF VAL FUNCTION 1 #TEXTVAL (A20)          /* ONLY CHECK FOR DATA NUMERICDIRECTION ------------------------------------------------------------------------------
READ #TEXTVAL(A20) IN ONEROW(1:1) FROM 1(1:20)  /* ONEROW(1:1)=(A1)(A19)(A1)  /* ONEROW(1:1)=(A1)(A19)(A1)  WRITE NOTITLE "IN ONEROW(1:1)" /* #TEXTVAL /* #TEXTVAL VAL  WRITE NOTITLE "VAL(#TEXTVAL)" /* #TEXTVAL /* #TEXTVAL VAL  ONLYONCE "TEXTVAL ONLY IS VALID" ONLYONCE  "TEXTVAL IS NOT IN NUMERICFORMAT" /* IS(N8)  "NOW TEXTVAL ONLY IS VALID"
END-DEFINE ------------------------------------------------------------------------------------------------------------------------------------------------------
```

Evaluation of a Logical Variable

```
operand1
```

Operand	Possible Structure					Possible Formats												Referencing Permitted	Dynamic Definition
Operand1	C	S	A											L				no	no

This option is used in conjunction with a logical variable (format L). A logical variable may take the value "TRUE" or "FALSE". As *operand1* you specify the name of the logical variable to be used.

```
Example of Logical Variable:
/* EXAMPLE "LOGLAV" /* EXAMPLE OF LOGICAL VARIABLE IN LOGICAL CONDITION /*------------------------------------------------------------------------------------------
DEFINE DATA LOCAL 1 #MATCH (L) INIT <TRUE> 1 #NUMBER (I4) END-DEFINE /*------------------------------------------------------------------------------------------
FOR #NUMBER 1 5  WRITE NOTITLE #MATCH (EM=VALUE:TRUE) IN "NUMBER =" #NUMBER  WRITE NOTITLE #MATCH (EM=VALUE:FALSE) IN "NUMBER =" #NUMBER IF #MATCH  MOVE FALSE TO #MATCH  ELSE  MOVE TRUE TO #MATCH  END-IF /*------------------------------------------------------------------------------------------
END-FOR /*------------------------------------------------------------------------------------------
TRUE    NUMBER = 1 IN    NUMBER = 1 FALSE    NUMBER = 2 OFF    NUMBER = 2 TRUE    NUMBER = 3 ON    NUMBER = 3 FALSE    NUMBER = 4 OFF    NUMBER = 4 TRUE    NUMBER = 5 ON    NUMBER = 5
```

Modified Control Variables

```
operand1 [NOT] MODIFIED
```

FIND	Possible Structure					Possible Formats												Referencing Permitted	Dynamic Definition
Operand1		S	A											C				no	no

 Copyright © Software AG 2002

This option is used to determine if the content of a field which has been assigned attributes dynamically has been modified during the execution of an INPUT statement.

Control variables referenced in an INPUT statement are always assigned the status "NOT MODIFIED" when the map is transmitted to the terminal.

Whenever the content of a field referencing a control variable is modified, the control variable is assigned the status "MODIFIED". When multiple fields reference the same control variable, the variable is marked "MODIFIED" if any of these fields is modified.

If *operand1* is an array, the result will be true if at least one of the array elements is modified (OR connection).

Note:
On mainframe computers, the profile parameter CVMIN (see the Natural Operations for Mainframes documentation) may be used to determine if a control variable is also to be set to "MODIFIED" if the value of the corresponding field is overwritten by an **identical** value.

SPECIFIED Option

parameter – name [NOT] SPECIFIED

This option is not available on mainframe computers.

This option is used to check whether an optional parameter in an invoked object (subprogram, external subroutine, dialog or ActiveX control) has received a value from the invoking object or not.

An optional parameter is a field defined with the keyword OPTIONAL in the DEFINE DATA PARAMETER statement of the invoked object. If a field is defined as OPTIONAL, a value can - but need not - be passed from an invoking object to this field.

In the invoking statement, the notation *n*X is used to indicate parameters for which no values are passed.

If you process an optional parameter which has not received a value, this will cause a runtime error. To avoid such an error, you use the SPECIFIED option in the invoked object to check whether an optional parameter has received a value or not, and then only process it if it has.

Parameter-name is the name of the parameter as specified in the DEFINE DATA PARAMETER statement of the invoked object.

For a field not defined as OPTIONAL, the SPECIFIED condition is always "TRUE".

Fields Used Within Logical Condition Criteria

Database fields and user-defined variables may be used to construct logical condition criteria. A database field which is a multiple-value field or is contained in a periodic group can also be used. If a range of values for a multiple-value field or a range of occurrences for a periodic group is specified, the condition is true if the search value is found in any value/occurrence within the specified range.

Each value used must be compatible with the field used on the opposite side of the expression. Decimal notation may be specified only for values used with numeric fields, and the number of decimal positions of the value must agree with the number of decimal positions defined for the field.

If the operands are not of the same format, the second operand is converted to the format of the first operand.

The following table shows which operand formats can be used together in a logical condition:

Operand2 → **↓ Operand1**	**A**	**B*n* (*n*≤4)**	**B*n* (*n*≥5)**	**D**	**T**	**I**	**F**	**L**	**N**	**P**	**GH**	**OH**
A	Y	Y	Y									
B*n* (*n*≤4)	Y	Y	Y			Y	Y		Y	Y		
B*n* (*n*≥5)	Y	Y	Y									
D				Y								
T					Y							
I		Y				Y	Y		Y	Y		
F		Y				Y	Y		Y	Y		
L												
N		Y				Y	Y		Y	Y		
P		Y				Y	Y		Y	Y		
GH											Y	
OH												Y

GH = GUI handle, OH = object handle.

If an array is compared with a scalar value, each element of the array will be compared with the scalar value. The condition will be true if at least one of the array elements meets the condition (OR connection).

If an array is compared with an array, each element in the array is compared with the corresponding element of the other array. The result is true only if all element comparisons meet the condition (AND connection).

See also Processing of Arrays.

An Adabas phonetic descriptor cannot be used within a logical condition.

Examples of Logical Condition Criteria:

```
FIND EMPLOYEES-VIEW WITH CITY = 'BOSTON' WHERE SEX = 'M' READ EMPLOYEES-VIEW BY NAME WHERE SEX = 'M' ACCEPT IF LEAVE-DUE GT 45 IF #A GT #B THEN COMPUTE #C = #A + #B REPEAT UNTIL #X = 500
```

Logical Operators in Complex Logical Expressions

Logical condition criteria may be combined using the Boolean operators "AND", "OR", and "NOT". Parentheses may also be used to indicate logical grouping.

The operators are evaluated in the following order:

1. **( )** Parentheses
2. **NOT** Negation
3. **AND** AND connection
4. **OR** OR connection

 Copyright © Software AG 2002

The following *logical-condition-criteria* may be connected by logical operators to form a complex *logical-expression*: relational expressions, extended relational expressions, MASK, SCAN, and BREAK options.

The syntax for a *logical-expression* is as follows:

```
[NOT] { logical-condition-criterion } [ { OR } logical-expression ]...
      { (logical-expression)        }   { AND }
```

Examples of Logical Expressions:

```
FIND STAFF-VIEW WITH CITY = 'TOKYO'              WHERE BIRTH GT 19610101 AND SEX = 'F'IF NOT (#CITY = 'A' THRU 'E')
```

For information on comparing arrays in a logical expression, see Processing of Arrays.

Note:
If multiple logical-condition-criteria are connected with "AND", the evaluation terminates as soon as the first of these criteria is not true.

Rules for Arithmetic Assignment

- Field Initialization
- Data Transfer
- Field Truncation and Field Rounding
- Result Format and Length in Arithmetic Operations
- Arithmetic Operations with Floating-Point Numbers
- Arithmetic Operations with Date and Time
- Performance Considerations for Mixed Format Expressions
- Precision of Results for Arithmetic Operations
- Error Conditions in Arithmetic Operations
- Processing of Arrays

Field Initialization

A field - user-defined variable or database field - which is to be used as an operand in an arithmetic operation must be defined with one of the following formats: N, P, I, F, D, T.

Note for reporting mode:
A field which is to be used as an operand in an arithmetic operation must have been previously defined.
A user-defined variable or database field used as a result field in an arithmetic operation need not have been previously defined.

All user-defined variables and all database fields defined in a DEFINE DATA statement are initialized to the appropriate zero or blank value when the program is invoked for execution.

Data Transfer

Data transfer is performed with a MOVE or COMPUTE statement. The following table summarizes data transfer compatibility and the rules for data transfer:

Sending Field	Receiving Field											
	N / P	**A**	**B*n* (*n* < 5)**	**B*n* (*n* > 4)**	**I**	**L**	**C**	**D**	**T**	**F**	**GH**	**OH**
N or **P**	Y	[2]	[3]	-	Y	-	-	-	Y	Y	-	-
A	-	Y	[1]	[1]	-	-	-	-	-	-	-	-
B*n* (*n* < 5)	[4]	[2]	[5]	[5]	Y	-	-	-	Y	Y	-	-
B*n* (*n* > 4)	-	[6]	[5]	[5]	-	-	-	-	-	-	-	-
I	Y	[2]	[3]	-	Y	-	-	-	Y	Y	-	-
L	-	[9]	-	-	-	Y	-	-	-	-	-	-
C	-	-	-	-	-	-	Y	-	-	-	-	-
D	Y	[9]	Y	-	Y	-	-	Y	[7]	Y	-	-
T	Y	[9]	Y	-	Y	-	-	[8]	Y	Y	-	-
F	Y	[9][10]	[3]	-	Y	-	-	-	Y	Y	-	-
GH	-	-	-	-	-	-	-	-	-	-	Y	-
OH	-	-	-	-	-	-	-	-	-	-	-	Y

Y	indicates data transfer compatibility.
-	indicates data transfer incompatibility.
[1]...	refers to the data conversion rules.

GH = GUI handle, OH = object handle.

See also Usage of Dynamic Variables.

Data Conversion

The following rules apply to converting data values:

1. **Alphanumeric to binary:** The value will be moved byte by byte from left to right. The result may be truncated or padded with trailing blank characters depending on the length defined and the number of bytes specified.
2. **(N,P,I) and binary (length 1-4) to alphanumeric:** The value will be converted to unpacked form and moved into the alphanumeric field left justified, i.e., leading zeros will be suppressed and the field will be filled with trailing blank characters. For negative numeric values, the sign will be converted to the hexadecimal notation "Dx". Any decimal point in the numeric value will be ignored. All digits before and after the decimal point will be treated as one integer value.
3. **(N,P,I,F) to binary (1-4 bytes):** The numeric value will be converted to binary (4 bytes). Any decimal point in the numeric value will be ignored (the digits of the value before and after the decimal point will be treated as an integer value). The resulting binary number will be positive or a two’s complement of the number depending on the sign of the value.
4. **Binary (1-4 bytes) to numeric:** The value will be converted and assigned to the numeric value right justified, i.e., with leading zeros. (Binary values of the length 1-3 bytes are always assumed to have a positive sign. For binary values of 4 bytes, the leftmost bit determines the sign of the number: 1=negative, 0=positive.) Any decimal point in the receiving numeric value will be ignored. All digits before and after

 Copyright © Software AG 2002

the decimal point will be treated as one integer value.

5. **Binary to binary:** The value will be moved from right to left byte by byte. Leading binary zeros will be inserted into the receiving field.
6. **Binary (>4 bytes) to alphanumeric:** The value will be moved byte by byte from left to right. The result may be truncated or padded with trailing blanks depending on the length defined and the number of bytes specified.
7. **Date (D) to time (T):** If date is moved to time, it is converted to time assuming time 00:00:00:0.
8. **Time (T) to date (D):** If time is moved to date, the time information is truncated, leaving only the date information.
9. **L,D,T,F to A:** The values are converted to display form and are assigned left justified.
10. If F is assigned to an alphanumeric field which is too short, the mantissa is reduced accordingly.

See also Usage of Large and Dynamic Variables/Fields.

Field Truncation and Field Rounding

The following rules apply to field truncation and rounding:

- High-order numeric field truncation is allowed only when the digits to be truncated are leading zeros. Digits following an expressed or implied decimal point may be truncated.
- Trailing positions of an alphanumeric field may be truncated.
- If the option ROUNDED is specified, the last position of the result will be rounded up if the first truncated decimal position of the value being assigned contains a value greater than or equal to 5.
  For the result precision of a division, see also Precision of Results for Arithmetic Operations.

Result Format and Length in Arithmetic Operations

The following table shows the format and length of the result of an arithmetic operation:

	I1	**I2**	**I4**	**N or P**	**F4**	**F8**
I1	I1	I2	I4	P*	F4	F8
I2	I2	I2	I4	P*	F4	F8
I4	I4	I4	I4	P*	F4	F8
N or P	P*	P*	P*	P*	F4	F8
F4	F4	F4	F4	F4	F4	F8
F8	F8	F8	F8	F8	F8	F8

P* is determined from the integer length and precision of the operands individually for each operation, as shown under Precision of Results for Arithmetic Operations.

The following decimal integer lengths and possible values are applicable for format I:

Format/Length	**Decimal Integer Length**	**Possible Values**
I1	3	-128 to 127
I2	5	-32 768 to 32 767
I4	10	-2 147 483 648 to 2 147 483 647

Arithmetic Operations with Floating-Point Numbers

Some General Considerations

Floating-point numbers (format F) are represented as a sum of powers of two (as are integer numbers (format I)), whereas unpacked and packed numbers (formats N and P) are represented as a sum of powers of ten.

In unpacked or packed numbers, the position of the decimal point is fixed. In floating-point numbers, however, the position of the decimal point (as the name indicates) is "floating", that is, its position is not fixed, but depends on the actual value.

Floating-point numbers are essential for the computing of trigonometric functions or mathematical functions such as sinus or logarithm.

The Precision of Floating-Point Numbers

Due to the nature of floating-point numbers, their precision is limited:

- For a variable of format/length F4, the precision is limited to approximately 7 digits.
- For a variable of format/length F8, the precision is limited to approximately 15 digits (16 digits on mainframe computers).

Values which have more significant digits cannot be represented exactly as a floating-point number. No matter how many additional digits there are before or after the decimal point, a floating-point number can cover only the leading 7 or 15 (16) digits respectively.

An integer value can only be represented exactly in a variable of format/length F4 if its absolute value does not exceed 2^{23} -1 (2^{24} -1 on mainframe computers).

Conversion to Floating-Point Representation

When an alphanumeric, unpacked numeric or packed numeric value is converted to floating-point format (for example, in an assignment operation), the representation has to be changed, that is, a sum of powers of ten has to be converted to a sum of powers of two.

Consequently, only numbers that are representable as a finite sum of powers of two can be represented exactly; all other numbers can only be represented approximately.

Examples:

This number has an exact floating-point representation:

$1.25 = 2^{0} + 2^{-2}$

This number is a periodic floating-point number without an exact representation:

$1.2 = 2^{0} + 2^{-3} + 2^{-4} + 2^{-7} + 2^{-8} + 2^{-11} + 2^{-12} + ...$

Thus, the conversion of alphanumeric, unpacked numeric or packed numeric values to floating-point values, and vice versa, can introduce small errors.

Platform-Dependent Differences

As already indicated by some of the differing limits mentioned above, the representation of floating-point numbers on mainframe computers is different from their representation on other platforms.

 Copyright © Software AG 2002

This explains why the same application, when run on different platforms, may return slightly different results when floating-point arithmetics are involved.

If you port a Natural application to another platform, also remember that the range of possible values for floating-point variables on a mainframe computer is different from that on other platforms:

- The possible value range for F4 and F8 variables on a mainframe computer is (approximately):
  $\pm 5.4 * 10^{-79}$ to $\pm 7.2 * 10^{75}$
- The possible value range on most other platforms is (approximately):
  for F4 variables: $\pm 1.17 * 10^{-38}$ to $\pm 3.40 * 10^{38}$
  for F8 variables: $\pm 2.22 * 10^{-308}$ to $\pm 1.79 * 10^{308}$

Note:
The representation used by your pocket calculator may also be different from the one used by your computer - which explains why results for the same computation may differ.

Arithmetic Operations with Date and Time

With formats D (date) and T (time), only addition and subtraction are allowed; multiplication and division are not allowed.

Date/time values can be added to/subtracted from one another; or integer values (no decimal digits) can be added to/subtracted from date/time values. Such integer values can be contained in fields of formats N, P, I, D, or T.

An integer value added to/subtracted from a date value is assumed to be in days. An integer value added to/subtracted from a time value is assumed to be in tenths of seconds.

For arithmetic operations with date and time, certain restrictions apply, which are due to the Natural's internal handling of arithmetic operations with date and time, as explained below.

Internally, Natural handles an arithmetic operation with date/time variables as follows:

COMPUTE *result-field = operand1 +/- operand2*

The above statement is resolved as:

1. *intermediate-result = operand1 +/- operand2*
2. *result-field = intermediate-result*

That is, in a first step Natural computes the result of the addition/subtraction, and in a second step assigns this result to the result field.

More complex arithmetic operations are resolved following the same pattern:

COMPUTE *result-field = operand1 +/- operand2 +/- operand3 +/- operand4*

The above statement is resolved as:

1. *intermediate-result1 = operand1 +/- operand2*
2. *intermediate-result2 = intermediate-result1 +/- operand3*
3. *intermediate-result3 = intermediate-result2 +/- operand4*
4. *result-field = intermediate-result3*

The internal format of such an *intermediate-result* depends on the formats of the operands, as shown in the tables below.

The following table shows the format of the *intermediate-result* of an **addition** (*intermediate-result = operand1 + operand2*):

Format of *operand1*	Format of *operand2*	Format of *intermediate-result*
D	**D**	**D***i*
D	**T**	**T**
D	**N, P, I**	**D**
T	**D, T, N, P, I**	**T**
N, P, I	**D**	**D**
N, P, I	**T**	**T**

The following table shows the format of the *intermediate-result* of a **subtraction** (*intermediate-result = operand1 - operand2*):

Format of *operand1*	Format of *operand2*	Format of *intermediate-result*
D	**D**	**D***i*
D	**T**	**T***i*
D	**N, P, I**	**D**
T	**D, T**	**T***i*
T	**N, P, I**	**T**
N, P, I	**D**	**D***i*
N, P, I	**T**	**T***i*

D*i* is a value in internal date format; **T***i* is a value in internal time format; such values can be used in further arithmetic date/time operations, but they cannot be assigned to a result field of format D (see the assignment table below).

In complex arithmetic operations in which an intermediate result of internal format **D***i* or **T***i* is used as operand in a further addition/subtraction, its format is assumed to be D or T respectively.

The following table shows which intermediate results can internally be assigned to which result fields (*result-field = intermediate-result*).

Format of *result-field*	Format of *intermediate-result*	Assignment possible
D	**D**, **T**	yes
D	**D***i*, **T***i*, **N**, **P**, **I**	*no*
T	**D**, **T**, **D***i*, **T***i*, **N**, **P**, **I**	yes
N, **P**, **I**	**D**, **T**, **D***i*, **T***i*, **N**, **P**, **I**	yes

A result field of format D or T must not contain a negative value.

 Copyright © Software AG 2002

Examples 1 and 2 (invalid):

```
COMPUTE DATE1 (D) = DATE2 (D) + DATE3 (D) COMPUTE DATE1 (D) = DATE2 (D) - DATE3 (D)
```

These operations are not possible, because the intermediate result of the addition/subtraction would be format D*i*, and a value of format D*i* cannot be assigned to a result field of format D.

Examples 3 and 4 (invalid):

```
COMPUTE DATE1 (D) = TIME2 (T) - TIME3 (T) COMPUTE DATE1 (D) = DATE2 (D) - TIME3 (T)
```

These operations are not possible, because the intermediate result of the addition/subtraction would be format T*i*, and a value of format T*i* cannot be assigned to a result field of format D.

Example 5 (valid):

```
COMPUTE DATE1 (D) = DATE2 (D) - DATE3 (D) + TIME3 (T)
```

This operation is possible. First, DATE3 is subtracted from DATE2, giving an intermediate result of format D*i*; then, this intermediate result is added to TIME3, giving an intermediate result of format T; finally, this second intermediate result is assigned to the result field DATE1.

If a format T value is assigned to a format D field, you must ensure that the time value contains a valid date component.

Performance Considerations for Mixed Format Expressions

When doing arithmetic operations, the choice of field formats has considerable impact on performance:

For business arithmetic under OpenVMS, UNIX and Windows, only fields of format I (integer) should be used.
If your computer is equipped with a math co-processor, format F (floating point) is faster than formats N or P and almost as fast as format I.
Without a math co-processor, format F is approximately as slow as formats N or P.

For business arithmetic on mainframe computers, only fields of format P (packed numeric) should be used. The number of decimal digits in all operands should agree where possible.

For scientific arithmetic, only fields of format F (floating point) should be used.

In expressions where formats are mixed between numeric (N, P) and floating point (F), a conversion to floating point format is performed. This conversion results in considerable CPU load. Therefore it is recommended to avoid mixed format expressions in arithmetic operations.

Precision of Results for Arithmetic Operations

Operation	Digits Before Decimal Point	Digits After Decimal Point
Addition/Subtraction	**Fi + 1** or **Si + 1** (whichever is greater)	**Fd** or **Sd** (whichever is greater)
Multiplication	**Fi + Si + 2**	**Fd + Sd** (maximum 7)
Division	**Fi + Sd**	(see below)
Exponentiation	**15 - Fd** (Exception: On mainframe computers, if the exponent has one or more digits after the decimal point, and on all other platforms in general, the exponentiation is internally carried out in floating point format. See Arithmetic Operations with Floating-Point Numbers for further information.)	**Fd**
Square Root	**Fi**	**Fd**
F = First operand **S** = Second operand **R** = Result **i** = Digits before decimal point **d** = Digits after decimal point		

Digits after Decimal Point for Division Results

The precision of the result of a division depends whether a result field is available or not:

- If a result field is available, the precision is: **Rd** or **Fd** (whichever is greater) *.
- If no result field is available, the precision is: **Fd** or **Sd** (whichever is greater) *.

A result field *is* available (or assumed to be available) in a COMPUTE and DIVIDE statement, and in a logical condition in which the division is placed after the comparison operator (for example: IF #A = #B / #C THEN ...). A result field is not (or not assumed to be) available in a logical condition in which the division is placed before the comparison operator (for example: IF #B / #C = #A THEN ...).

Exception: If both dividend and divisor are of integer format and at least one of them is a variable, the division result is always of integer format (regardless of the precision of the result field and of whether the ROUNDED option is used or not).

* If the ROUNDED option is used, the precision of the result is internally increased by one digit before the result is actually rounded.

Error Conditions in Arithmetic Operations

In an addition, subtraction, multiplication or division, an error occurs if the total number of digits (before and after the decimal point) of the result is greater than 31.

In an exponentiation, an error occurs in any of the following situations:

- if the base is of packed format and either the result has over 16 digits or any intermediate result has over 15 digits;
- if the base is of floating-point format and the result is greater than approximately $7 * 10^{75}$.

 Copyright © Software AG 2002

Processing of Arrays

Generally, the following rules apply:

- All scalar operations may be applied to array elements which consist of a single occurrence.
- If a variable is defined with a constant value (for example, #FIELD (I2) CONSTANT <8>), the value will be assigned to the variable at compilation, and the variable will be treated as a constant. This means that if such a variable is used in an array index, the dimension concerned has a *definite* number of occurrences.
- If an assignment/comparison operation involves two arrays with a different number of dimensions, the "missing" dimension in the array with fewer dimensions is assumed to be (1:1).

Example:

If **#ARRAY1 (1:2)** is assigned to **#ARRAY2 (1:2,1:2)**,
#ARRAY1 is assumed to be **#ARRAY1 (1:1,1:2)**.

Assignment Operations with Arrays

If an array range is assigned to another array range, the assignment is performed element by element.

If a single occurrence is assigned to an array range, each element of the range is filled with the value of the single occurrence. (For a mathematical function, each element of the range is filled with the result of the function.)

Before an assignment operation is executed, the individual dimensions of the arrays involved are compared with one another to check if they meet one of the conditions listed below. The dimensions are compared independently of one another; that is, the 1st dimension of the one array is compared with the 1st dimension of the other array, the 2nd dimension of the one array is compared with the 2nd dimension of the other array, and the 3rd dimension of the one array is compared with the 3rd dimension of the other array.

The assignment of values from one array to another is only allowed under one of the following conditions:

- The number of occurrences is the same for both dimensions compared.
- The number of occurrences is indefinite for both dimensions compared.
- The dimension that is assigned to another dimension consists of a single occurrence.

The following program shows which array assignment operations are possible.

Example - Array Assignments:

Comparison Operations with Arrays

Generally, the following applies: if arrays with multiple dimensions are compared, the individual dimensions are handled independently of one another; that is, the 1st dimension of the one array is compared with the 1st dimension of the other array, the 2nd dimension of the one array is compared with the 2nd dimension of the other array, and the 3rd dimension of the one array is compared with the 3rd dimension of the other array.

The comparison of two array dimensions is only allowed under one of the following conditions:

- The array dimensions compared with one another have the same number of occurrences.
- The array dimensions compared with one another have an indefinite number of occurrences.
- All array dimensions of one of the arrays involved are single occurrences.

The following program shows which array comparison operations are possible:

Example - Array Comparisons:

When you compare two array ranges, note that the following two expressions lead to different results:

```
#ARRAY1(*) NOT EQUAL #ARRAY2(*)
 NOT #ARRAY1(*) = #ARRAY2(*)
```

Example:

Condition A:

```
IF #ARRAY1(1:2) NOT EQUAL #ARRAY2(1:2)
```

This is equivalent to:

```
IF (#ARRAY1(1) NOT EQUAL #ARRAY2(1)) AND (#ARRAY1(2) NOT EQUAL #ARRAY2(2))
```

Condition A is therefore true if the first occurrence of #ARRAY1 does not equal the first occurrence of #ARRAY2 *and* the second occurrence of #ARRAY1 does not equal the second occurrence of #ARRAY2.

Condition B:

```
IF NOT #ARRAY1(1:2) = #ARRAY2(1:2)
```

This is equivalent to:

```
IF NOT (#ARRAY1(1)= #ARRAY2(1) AND #ARRAY1(2) = #ARRAY2(2))
```

This in turn is equivalent to:

```
IF (#ARRAY1(1) NOT EQUAL #ARRAY2(1)) OR (#ARRAY1(2) NOT EQUAL #ARRAY2(2))
```

Condition B is therefore true if *either* the first occurrence of #ARRAY1 does not equal the first occurrence of #ARRAY2 *or* the second occurrence of #ARRAY1 does not equal the second occurrence of #ARRAY2.

Arithmetic Operations with Arrays

In arithmetic operations (in COMPUTE, ADD or MULTIPLY statements), array ranges may be used in the following ways:

- *range + range = range*.
  The range dimensions must be equal.
  The addition is performed element by element.
- *range * range = range*.
  The range dimensions must be equal.
  The multiplication is performed element by element.
- *scalar + range = range*.
  The range dimensions must be equal.
  The scalar is added to each element of the range.
- *range * scalar = range*.
  The range dimensions must be equal.
  Each element of the range is multiplied by the scalar.
- *range + scalar = scalar*.
  Each element of the range is added to the scalar and the result is assigned to the scalar.
- *scalar * range = scalar2*.
  The scalar is multiplied by each element of the array and the result is assigned to scalar2.

 Copyright © Software AG 2002

System Variables

This section describes the Natural system variables. System variables are used to display system information. They may be referenced at any point within a Natural program.

- Alphabetical List of Variables
- Date and Time System Variables

Content modifiable:
This indicates whether in a Natural program you can assign another value to the system variable, that is, overwrite its content as generated by Natural.

See also:

- Large and Dynamic Variables/Fields (in the Natural for Windows documentation).

Alphabetical List of Variables

The following system variables are available:

*APPLIC-ID	*ERROR-NR	*LIBRARY-ID	*PARM-USER	*USER
*APPLIC-NAME	*ERROR-TA	*LINE-COUNT	*PATCH-LEVEL	*USER-NAME
*COM	*ETID	*LINESIZE	*PF-KEY	*WINDOW-LS
*CONTROL	*EVENT	*LOG-LS	*PF-NAME	*WINDOW-POS
*CONVID	*GROUP	*LOG-PS	*PID	*WINDOW-PS
*CPU-TIME	*HARDCOPY	*MACHINE-CLASS	*PROGRAM	*WINMGR
*COUNTER	*HARDWARE	*NATVERS	*ROWCOUNT	*WINMGRVERS
*CURS-COL	*HOSTNAME	*NET-USER	*SCREEN-IO	
*CURS-FIELD	*INIT-ID	*NUMBER	*SERVER-TYPE	
*CURS-LINE	*INIT-PROGRAM	*OCCURRENCE	*STARTUP	
*CURSOR	*INIT-USER	*OPSYS	*STEPLIB	
*DATA	*ISN	*OS	*SUBROUTINE	
*DEVICE	*LANGUAGE	*OSVERS	*THIS-OBJECT	
*DIALOG-ID	*LENGTH	*PAGESIZE	*TPSYS	
*ERROR-LINE	*LEVEL	*PAGE-NUMBER	*UI	

*APPLIC-ID

Format/length:	A8
Content modifiable:	No

This system variable contains the ID of the library to which the user is currently logged on.

*APPLIC-NAME

Format/length:	A32
Content modifiable:	No

If Natural Security is installed, this system variable contains the name of the library to which the user is logged on.

If Natural Security is not installed, this system variable contains the name "SYSTEM".

*COM

Format/length:	A128
Content modifiable:	Yes

This system variable designates a communication area which can be used to process data from outside a screen window.

Normally when a window is active, no data can be entered on the screen outside the window. However, if a map contains *COM as a modifiable field, that field will still be available for the user to enter data when a window is currently on the screen. Further processing can then be made dependent on the content of *COM. This allows you to implement user interfaces where a user can always enter data in the command line, even when a window with its own input fields is active.

Note:
Although *COM can be used as a modifiable field in an INPUT statement, it is **not** treated as an input field, but as a system variable; that is, any input entered into the *COM field will be taken as it is, without any input processing (e.g. conversion to upper case) being performed on it.
Once *COM has been displayed on the screen via an INPUT statement, every subsequent INPUT or REINPUT statement will cause the current content of *COM to be displayed.

 Copyright © Software AG 2002

*CONTROL

This system variable is only available under Windows 98 and Windows NT/2000.

Format/length:	Handle
Content modifiable:	No

This system variable contains the handle of the dialog element for which the current event has been triggered.

For details on events, see Event-Driven Programming Techniques in the Natural User's Guide for Windows.

*CONVID

Format/length:	I4
Content modifiable:	Yes

This system variable contains the conversation ID of the current conversational remote procedure call (RPC). This ID is set by an OPEN CONVERSATION statement.

Via an OPEN CONVERSATION statement, a client can get a server for exclusive use to execute a number of services (subprograms) within one server process. This exclusive use is called conversation. The OPEN CONVERSATION statement is used to open a conversation and specify the subprograms to be involved in this conversation. When an OPEN CONVERSATION statement is executed, it assigns a unique ID which identifies the conversation to the system variable *CONVID.

Several conversations can be open at the same time. To switch from one open conversation to another, you assign the corresponding conversation ID to *CONVID.

For further information on Natural RPC, see the Natural RPC documentation.

*COUNTER

Format/length:	P10
Content modifiable:	Yes

This system variable contains the number of times a processing loop initiated by a FIND, READ or HISTOGRAM statement has been entered.

(r) in parentheses after *COUNTER is used to indicate the statement label or source-code line number of the FIND, READ, or HISTOGRAM. If (r) is not specified, *COUNTER represents the number of times the currently active processing loop has been entered.

*COUNTER is not incremented if a record is rejected as a result of the criteria specified in a WHERE clause. *COUNTER is incremented if a record is rejected as a result of an ACCEPT/REJECT statement.

*CPU-TIME

Format/length:	I4
Content modifiable:	No

This system variable is not available on mainframe computers.

*CPU-TIME contains the cpu time currently used by the Natural process.
Warning: Due to cpu clock restrictions on some UNIX systems, the value may wrap around after accumulating only 4295 seconds of CPU time (about 72 minutes).

*CURS-COL

Format/length:	P3
Content modifiable:	Yes (however, a negative value must not be assigned)

This system variable contains the number of the column in which the cursor is currently positioned.

The cursor position is defined within the currently active window, regardless of its physical placement on the screen, starting with position 1/1 from the upper left corner of a logical page.

If the value of *CURS-COL is negative, this indicates that the cursor is outside the active window. If *CURS-COL is negative, *CURS-LINE will also contain a negative value. In this case, the absolute values of both system variables indicate the position of the cursor on the physical screen.

Note:
The message line, function-key lines and infoline/statistics line are not counted as data lines on the screen.

*CURS-FIELD

Format/length:	I4
Content modifiable:	No

This system variable contains the internal identification of the field in which the cursor is currently positioned.

*CURS-FIELD cannot be used by itself, but only in conjunction with the POS function. You may use them to check if the cursor is currently positioned in a specific field and have processing performed depending on that condition. See the POS function for details.

If the cursor is not in a field or if no REINPUT is possible, *CURS-FIELD contains 0.

Note:
The value of *CURS-FIELD serves only as internal identification of the field and cannot be used for arithmetic operations.

 Copyright © Software AG 2002

*CURS-LINE

Format/length: P3

Content modifiable: Yes (however, a negative value or 0 must not be assigned)

This system variable contains the number of the line in which the cursor is currently positioned.

The cursor position is defined within the current active window, regardless of its physical placement on the screen, starting with position 1/1 from the upper left corner of a *logical* page.

Note:
The message line, function-key lines and infoline/statistics line are not counted as data lines on the screen.

*CURS-LINE may also contain one of the following values:

Value	Cursor Position
0	On the top or bottom horizontal frame line of a window.
-1	On the Natural message line.
-2	On the Natural infoline/statistics line.
-3	On the upper function-key (number) line.
-4	On the lower function-key (name) line.

If the value of *CURS-COL is negative, which indicates that the cursor is outside the active window, *CURS-LINE will also contain a negative value. In this case, the *absolute* values of both system variables indicate the position of the cursor on the *physical* screen.

*CURSOR

Format/length: N6

Content modifiable: No

This system variable contains the position of the cursor on the input screen at the time the ENTER key or a function key is pressed.

Note:
Instead of *CURSOR, it is recommended that the system variables *CURS-LINE and *CURS-COL be used. *CURSOR only continues to be available for compatibility with previous Natural versions.

*DATA

Format/length: N3

Content modifiable: No

This system variable contains the number of data elements in the Natural stack which are available to the next INPUT statement as input data. *DATA will contain "0" when the stack is empty. A value of "-1" indicates the next element in the stack is a command or the name of a Natural transaction.

The settings of the IA and ID parameters at the time of execution of the STACK statement are used to determine the *DATA value.

*DEVICE

Format/length: A8

Content modifiable: No

This system variable contains the device type/mode from which Natural has been invoked. It may contain one of the following values:

Value	Description
BATCH	Batch mode.
COLOR	3279 compatibility. 3278 screen device (device with extended attribute support).
VIDEO	3270 screen device, PC screen device, DEC VT terminal or any type of UNIX terminal.
TTY	Teletype or other start/stop device.
PC	Usage of Natural Connection has been activated (by profile parameter PC=ON or terminal command %+).
BTX	BTX device.
SPOOL	3270 printer device.
ASYNCH	Asynchronous session.

 Copyright © Software AG 2002

*DIALOG-ID

This system variable is only available under Windows 98 and Windows NT/2000.

Format/length:	I4
Content modifiable:	No

This system variable contains the ID of the current instance of a dialog.

For details on dialogs, see Event-Driven Programming Techniques in the Natural User's Guide for Windows.

*ERROR-LINE

Format/length:	N4
Content modifiable:	No

This system variable contains the source-code line number of the statement that caused an error.

*ERROR-NR

Format/length:	N7
Content modifiable:	Yes

This system variable contains the error number of the error which caused an ON ERROR condition to be entered.

Normally, *ERROR-NR contains the Natural *system* error number which caused an error condition to be entered; however, when a "REINPUT WITH TEXT **nnnn*" statement is executed, the *application-specific* message number *nnnn* is placed into *ERROR-NR.

You may modify the content of this system variable via a Natural program; however, not within an ON ERROR statement block.

*ERROR-NR is reset to "0" when another level 1 program is executed.

*ERROR-TA

Format/length: A8

Content modifiable: Yes

This system variable contains the name of the program which is to receive control in the event of an error condition.

When an error occurs, Natural will execute a STACK TOP DATA statement and place at the top of the stack the following information, which can be used as INPUT data by an error transaction: Error number (N4 if SG=OFF; N5 if SG=ON), Line number (N4), Status (A1), Program name (A8), Level (N2). If the Status is either "C" or "L", the Line number will be "0". The Status may be one of the following:

C	Command processing error.
L	Logon error.
O	Object time error.
S	Non-correctable Syntax error.
R	Error on Remote server (in conjunction with Natural RPC).

*ETID

Format/length: A8

Content modifiable: No

This system variable contains the current identifier of transaction data for Adabas. The default value is one of the following:

- the value of the Natural profile parameter ETID,
- the user ID as passed from the TP monitor (on mainframe computers only),
- the value provided in the user exit during Natural initialization (on mainframe computers only),
- the value from the security profile of the currently active user (applies only under Natural Security).

*EVENT

This system variable is only available under Windows 98 and Windows NT/2000.

Format/length: A32

Content modifiable: No

This system variable contains the name of the current event.

For details on events, see Event-Driven Programming Techniques in the Natural User’s Guide for Windows.

 Copyright © Software AG 2002

*GROUP

Format/length: A8

Content modifiable: No

This system variable is applicable under Natural Security only. It contains the ID via which a user is logged on to a protected library, that is, the ID via which the user is linked to the library. This may be either the ID of the group via which the user is linked or the user's own ID (if he or she is linked directly).

In the case of a logon to an unprotected library (where no link is used), *GROUP will be blank.

If Natural Security is not active, *GROUP will be blank.

*HARDCOPY

Format/length: A8

Content modifiable: Yes

This system variable contains the name of the hardcopy device which will be used when the terminal command %H is used.

*HARDWARE

Format/length: A16

Content modifiable: No

This system variable contains the name of the hardware platform on which Natural is running (for example, AT&T). This value is supplied by the operating system.

*HOSTNAME

Format/length: A64

Content modifiable: No

This system variable is not available on mainframe computers.

The name of the machine Natural runs on, the same as returned by the OS-command "hostname".

*INIT-ID

Format/length:	A8
Content modifiable:	No

Under OpenVMS and UNIX

*INIT-ID contains the ID of the device from which Natural was invoked.

Under Windows 98 and Windows NT/2000

*INIT-ID contains the value "PC_WIN".

On mainframe computers

*INIT-ID contains the terminal ID (defined according to the conventions of the TP system) of the terminal from which Natural was invoked.

On mainframe computers in batch mode

*INIT-ID contains the step name of the Natural job.

For an asynchronous Natural session under Com-plete or UTM

*INIT-ID contains the terminal ID of the task that has started the asynchronous session.

For an asynchronous session under CICS

*INIT-ID contains the CICS task number of the asynchronous task.

*INIT-PROGRAM

Format/length:	A8
Content modifiable:	No

This system variable contains the name of the program (transaction) that is currently executing as Natural.

In batch mode under OS/390

*INIT-PROGRAM contains the name of the job under which the Natural session is running.

Under OpenVMS, UNIX and Windows

*INIT-PROGRAM contains the value "Natural".

*INIT-USER

Format/length:	A8
Content modifiable:	No

This system variable contains the user ID of the user.

 Copyright © Software AG 2002

Under OpenVMS and UNIX and Windows

*INIT-USER contains the value of the profile parameter USER in the parameter file used.
If no value is specified for the USER parameter, *INIT-USER contains the user ID used to log in to OpenVMS or UNIX, or under Windows the ID you were requested to enter when starting Natural (default value: "SAGPC").

Under UTM

*INIT-USER contains the user ID defined for the UTM application; if no user IDs are defined for the UTM application, *INIT-USER is identical to *INIT-ID.

Under TIAM

The value of *INIT-USER is determined by the parameter USERID in macro NATTIAM: If USERID=USER or NO (default), it contains the BS2000/OSD job name specified with the LOGON command; if no BS2000/OSD job name has been specified, *INIT-USER contains the same as with USERID=SYSTEM (or YES), that is, the BS2000/OSD user ID.

In batch mode on mainframe computers

*INIT-USER contains name of the job under which the Natural session is running.

In batch mode under OS/390

The value of *INIT-USER is determined by the parameter USERID in the Natural/OS/390 batch interface (macro NTOS): If USERID=YES, the value will be taken from the security access control block (ACEE) of the security package (for example, RACF or ACF2) being used. If no security package is used, the value will be taken from the USER parameter in the job card. If no USER parameter is specified, the value will be same as with USERID=NO, that is, the name of the job under which the Natural session is running.

*ISN

Format/length:	P10
Content modifiable:	Yes

This system variable contains the Adabas internal sequence number (ISN) of the record currently being processed within a processing loop initiated by a FIND or READ statement.

`(r)` in parentheses after *ISN is used to indicate the label or statement number of the statement in which the FIND or READ was issued. If `(r)` is not specified, *ISN represents the ISN of the record currently being processed in the currently active processing loop.

For the HISTOGRAM statement, *ISN contains the number of the occurrence in which the descriptor value last read is contained (*ISN = 0 if the descriptor is not contained within a periodic group).

*ISN for DL/I and SQL Databases

*ISN cannot be used for DL/I and SQL databases.

*ISN for VSAM

For VSAM databases, *ISN can only be applied to ESDS and RRDS. For ESDS, *ISN contains the relative byte address (RBA) and for RRDS the relative record number (RRN) of the record currently being processed within a processing loop initiated by a FIND or READ statement.

*LANGUAGE

Format/length: I1

Content modifiable: Yes

This system variable contains the language indicator (language code). This language indicator is used for edit masks of date fields, Natural error messages and user error messages as used in the statements INPUT and REINPUT.

A one-character code is assigned to each language code; this one-character code is used to replace all "&" in names of language-specific objects (for example, maps, dialogs, helproutines, subprograms).

You can specify up to 60 different language codes. The codes are listed on the following pages.

For details on how to use language codes, see also **Designing User Interfaces** in the *Natural User's Guide for Mainframes*.

The following languages are assigned to the individual language codes (the right-hand column shows the corresponding one-character codes to be used in names of language-specific maps etc.):

Code	Language	Code in Map Names
Left-to-right single-byte languages with Latin lower case:		
1	English	1
2	German	2
3	French	3
4	Spanish	4
5	Italian	5
6	Dutch	6
7	Turkish	7
8	Danish	8
9	Norwegian	9
10	Albanian	A
11	Portuguese	B
12	Chinese Latin (Taiwan)	C
13	Czech	D
14	Slovak	E
15	Finnish	F
16	Hungarian	G
17	Icelandic	H
18	Korean	I
19	Polish	J
20	Romanian	K
21	Swedish	L

 Copyright © Software AG 2002

Code	Language	Code in Map Names
22	Croatian	M
23	Catalan	N
24	Basque	O
25	Afrikaans	P
Left-to-right single-byte languages without Latin lower case:		
26	Bulgarian	Q
27	Greek	R
28	Japanese (Katakana)	S
29	Russian	T
30	Serbian	U
Bi-directional single-byte languages without Latin lower case:		
31	Arabic	V
32	Farsi (Iran)	W
33	Hebrew	X
34	Urdu (Pakistan)	Y
35	(reserved for future use)	Z
36	(reserved for future use)	a
37	(reserved for future use)	b
38	(reserved for future use)	c
39	(reserved for future use)	d
40	(reserved for future use)	e
User-assigned languages:		
41	(free for you to assign a language)	f
42	(free for you to assign a language)	g
43	(free for you to assign a language)	h
44	(free for you to assign a language)	i
45	(free for you to assign a language)	j
46	(free for you to assign a language)	k
47	(free for you to assign a language)	l
48	(free for you to assign a language)	m
49	(free for you to assign a language)	n
50	(free for you to assign a language)	o
Multiple-byte languages:		
51	Hindi	p
52	Malayan	q

Copyright © Software AG 2002

Code	Language	Code in Map Names
53	Thai	r
54	(reserved for future use)	s
55	(reserved for future use)	t
56	(reserved for future use)	u
Double-byte languages:		
57	Chinese (People's Republic of China)	v
58	Chinese (Republic of China)	w
59	Japanese (Kanji)	x
60	Korean	y

 Copyright © Software AG 2002

*LENGTH

Format/length:	I4
Content modifiable:	No

This system variable is not available on mainframe computers.

This system variable returns the used size of a dynamic Natural field or variable in bytes. *LENGTH may be used only for dynamic variables to get the currently used size.

*LEVEL

Format/length:	N2
Content modifiable:	No

This system variable contains the level number of the program, subprogram, external subroutine, map, helproutine or dialog which is currently active. Level 1 is a main program.

*LEVEL does not apply to inline subroutines.

*LIBRARY-ID

Format/length:	A8
Content modifiable:	No

This system variable contains the current library ID (as specified by the user in the LOGON command).

This variable is the equivalent of the variable *APPLIC-ID.

*LINE-COUNT

Format/length:	P5
Content modifiable:	No

This system variable contains the line number of the current line within the current page.

This variable is used by Natural to determine the line number for the next line of the report.

The value of *LINE-COUNT is incremented by "1" for each line to be output. The value is updated during the execution of a WRITE, SKIP, DISPLAY, PRINT or INPUT statement and contains the number of the last line on the page that has been output.

An EJECT or NEWPAGE statement causes *LINE-COUNT to be reset to "1" (except in the case of NEWPAGE WITH TITLE, where the value of *LINE-COUNT depends on the number of lines output as title).

The maximum line number permitted is 250.

If multiple reports are being produced by the program, (rep) in parentheses after *LINE-COUNT is used to specify the report identification for which the current line number is being requested.

*LINESIZE

Format/length: N7

Content modifiable: No

This system variable contains the physical line size of the I/O device from which Natural was invoked (if the TP system is able to provide such).

*LOG-LS

Format/length: N3

Content modifiable: No

This system variable contains the line size of the logical page that is output with the primary report.

*LOG-LS is only applicable to the primary report, not to any additional report.

*LOG-PS

Format/length: N3

Content modifiable: No

This system variable contains the page size of the logical page that is output with the primary report.

*LOG-PS is only applicable to the primary report, not to any additional report.

*MACHINE-CLASS

Format/length: A16

Content modifiable: No

This system variable contains the name of the machine class on which Natural is running.

It can contain one of the following values:

MAINFRAME
PC
UNIX
VMS

 Copyright © Software AG 2002

*NATVERS

Format/length:	A8
Content modifiable:	No

This system variable is not available on mainframe computers.

This system variable contains the Natural version excluding the patch level.

*NET-USER

Format/length:	A253
Content modifiable:	No

This system variable contains the complete user ID of an authenticated client request. The ID consists of the domain name and the actual user ID.

By default, the value of *NET-USER is identical to the one of *USER.

When a NaturalX server receives an authenticated request, the user ID of this request is passed to the server and placed into *NET-USER. (The DCOM function "CoQueryClientBlanket" is used for this purpose.)

After the NaturalX server has processed the request, *NET-USER is reset to the value it contained before the request (that is, the value of *USER).

A request which is not authenticated has no effect on *NET-USER.

In a non-NaturalX server environment, the content of *NET-USER is always the same as that of *USER.

*NUMBER

Format/length:	P10
Content modifiable:	Yes

This system variable contains either the number of records which were selected as a result of a FIND statement (as a result of the WITH clause) or the number of values selected as a result of a HISTOGRAM statement.

(r) in parentheses after *NUMBER is used to indicate the statement label or source-code line number of the statement in which the FIND or HISTOGRAM statement was issued. If *(r)* is not specified, *NUMBER represents the number of records selected from the FIND or HISTOGRAM used to initiate the currently active processing loop.

Note:
If *NUMBER contains the value "99999999999", this indicates that the Adabas file accessed is protected by the Adabas facility Security By Value.

*NUMBER for DL/I

For DL/I databases, *NUMBER does not contain the number of segment occurrences found. It contains "0" if no segment occurrence satisfies the search criterion, and it contains a value of 8,388,607=X’7FFFFF’ if at least one segment occurrence satisfies the search criterion.

*NUMBER for SQL Databases

When used with a FIND NUMBER without a WHERE clause or with a HISTOGRAM statement, *NUMBER will contain the number of rows found. Otherwise, when applied to an SQL database table, *NUMBER will not contain the number of rows found: *NUMBER will be "0" if no rows have been found; any value other than "0" indicates that rows have been found, but the value will have no relation to the number of rows actually found.

If a FIND NUMBER with a WHERE clause is used, the number of rows found is returned in *COUNTER.

*NUMBER for VSAM

For VSAM databases, *NUMBER only contains the number of records found, when used with a HISTOGRAM statement, or with a FIND statement which uses the operator "EQUAL TO" in the search criterion. With any other operator, *NUMBER will not contain the number of records found: *NUMBER will be "0" if no records have been found; any other value indicates that records have been found, but the value will have no relation to the number of records actually found.

*OCCURRENCE

Format/length:	I4
Content modifiable:	No

This system variable contains the current number of occurrences of a variable array.

In a parameter data area, you can use the index notation "1:V" to define an array with a variable number of occurrences (see the DEFINE DATA statement in the Natural Statements documentation). The actual number of occurrences of such a variable array is determined at runtime. With *OCCURRENCE, you can ascertain the actual number of array occurrences.

*OCCURRENCE is evaluated at runtime when the subprogram/subroutine in which the system variable is specified is executed.

After *OCCURRENCE you specify (in parentheses) the name by which the array concerned is defined in the Parameter Data Area:

```
*OCCURRENCE(#ARRAY)
```

After the name, you can specify the number *n* of the dimension (1, 2, 3 or *) whose number of occurrences you wish to ascertain:

```
*OCCURRENCE(#ARRAY,n)
```

An asterisk (*) means that the number of occurrences of **all** dimensions is ascertained.

If you specify no dimension, the 1st dimension will be assumed.

Example:

```
DEFINE DATA   PARAMETER     1 #ARRAY (A5/1:V)   LOCAL     1 #I (I4)     ... END-DEFINE ... FOR #I = 1 TO *OCCURRENCE(#ARRAY)   ... END-FOR ...
```

Note:
*OCCURRENCE may be abbreviated as *OCC.

 Copyright © Software AG 2002

For further examples, see programs/subprograms OCC1P and OCC1N, as well as OCC2P and OCC2N in library SYSEXRM.

*OPSYS

Format/length: A8

Content modifiable: No

This system variable contains the name of the operating system that is being used.

It can contain one of the following values:

ATT_OSX	**FACOM/XA**	**RS_6000**
AVIION	**FUJI M73**	**SCO**
BS2000	**HP_HPUX**	**SINIX_52**
BS2/XS	**MSDOS**	**SINIX_54**
BULL/BOS	**MS_OS/2**	**SUN_SOLA**
CMS	**MVS/ESA**	**SUN_SUNO**
CMS/XA	**MVS/XA**	**UNISYS 5**
DEC-OSF/	**NCR 3000**	**UNISYS 6**
DOS/VS	**OS**	**VSE/ESA**
DPS300	**OS/400**	**WNT-AXP**
DRS 6000	**OVMS/AXP**	**WNT-X86**
FACOM	**OVMS/VAX**	

Note:
Instead of *OPSYS, it is recommended that the system variables *MACHINE-CLASS, *HARDWARE and *OS be used, as they allow a more precise distinction of the environment in which Natural is running.

*OS

Format/length: A32

Content modifiable: No

This system variable contains the name of the operating system under which Natural is running (for example, HP-UX). This value is supplied by the operating system.

*OSVERS

Format/length: A16

Content modifiable: No

This system variable contains the version number of the operating system under which Natural is running. This value is supplied by the operating system.

*PAGESIZE

Format/length:	N7
Content modifiable:	No

This system variable contains the physical page size of the I/O device from which Natural was invoked (if the TP system is able to provide such).

*PAGE-NUMBER

Format/length:	P5
Content modifiable:	Yes

This system variable contains the current value for page number of an output report.

If multiple reports are being produced by the program, *(rep)* in parentheses after *PAGE-NUMBER is used to specify the report identification for which the current page number is being requested.

This variable is defined by Natural at the time formatting for the report is started. Therefore, the parameter has no meaning until the first FORMAT, WRITE, or DISPLAY statement for any given report has been issued. This variable may be modified by a Natural program.

This variable is used by Natural to determine the page number for the next page of the report. The value is always incremented by 1 for the next page initiated by WRITE, DISPLAY, SKIP or NEWPAGE statements. EJECT does not cause *PAGE-NUMBER to be incremented.

 Copyright © Software AG 2002

*PARM-USER

Format/length: A253

Content modifiable: No

This system variable is not available on mainframe computers.

This system variable contains the name of the parameter file currently in use.

*PATCH-LEVEL

Format/length: A8

Content modifiable: No

This system variable is not available on mainframe computers.

This system variable contains the current patch-level number as a string value.

*PF-KEY

Format/length: A4

Content modifiable: No

This system variable contains the identification of the key which was pressed last.

*PF-KEY can contain one of the following values:

PA1 to **PA3**	Program Attention keys 1 to 3
PF1 to **PF48**	Program Function keys 1 to 48
ENTR	ENTER key
CLR	CLEAR key
PEN	Light pen
PGDN	PAGE DOWN key (only under OpenVMS and UNIX).
PGUP	PAGE UP key (only under OpenVMS and UNIX).

*PF-KEY only contains the identification of a key if that key has been made sensitive on that level; otherwise *PF-KEY will contain "ENTR".

Note:
When you compare the content of *PF-KEY with a range of values, remember that *PF-KEY contains an alphanumeric value.

*PF-NAME

Format/length:	A10
Content modifiable:	No

This system variable contains the name of the function key that was pressed last, that is, the name as assigned to the key with the NAMED clause of the SET KEY statement.

This allows you to perform processing depending on a specific function name, not a specific key. For example, if you wish to allow users to invoke help by pressing either PF1 or PF13, you assign the name "HELP" to the keys PF1 and PF13 and make the invoking of help dependent on *PF-NAME='HELP': the help will then be invoked no matter whether the user presses PF1 or PF13 to invoke it.

*PID

Format/length:	A32
Content modifiable:	No

This system variable is not available on mainframe computers.

This system variable contains the current process ID as a string value.

*PROGRAM

Format/length:	A8
Content modifiable:	No

This system variable contains the name of the Natural object that is currently being executed.

*ROWCOUNT

Format/length:	I4
Content modifiable:	No

This system variable contains the number of rows that were deleted, updated or inserted by one of the Natural SQL statements "searched" DELETE, "searched" UPDATE or INSERT (with *select-expression*) respectively. *ROWCOUNT always refers to the last executed one of these statements.

 Copyright © Software AG 2002

*SCREEN-IO

Format/length: L

Content modifiable: No

This system variable indicates whether a screen I/O is possible or not.

It can contain one of the following values:

- TRUE = screen I/O is possible,
- FALSE = screen I/O is not possible.

In an interactive Natural session, *SCREEN-IO is initialized with "TRUE".

If Natural was started as a DB2 Stored Procedures server (*SERVER-TYPE = DB2-SP) or as RPC server (*SERVER-TYPE = RPC) *SCREEN-IO is set to "FALSE".

If Natural was started as DCOM server (*SERVER-TYPE = DCOM), *SCREEN-IO is set to "FALSE", while the server is executing a method called by COM/DCOM.

When *SCREEN-IO is set to "FALSE" and a statement which requires user interaction is executed, Natural issues error NAT0723.

*SERVER-TYPE

Format/length: A32

Content modifiable: No

This system variable indicates the server type Natural has been started as.

It can contain one of the following values:

DB2-SP	Natural DB2 Stored Procedures server.
DCOM	NaturalX DCOM server.
DEVELOP	Natural development server.
RPC	Natural RPC server.

If Natural is not started as a server, *SERVER-TYPE is set to blanks.

Note:
*SERVER-TYPE refers to Natural as a whole, **not** to the Natural program currently being executed (which may run as a client program or as a server program within a server Natural).

*STARTUP

Format/length:	A8
Content modifiable:	Yes

The program whose name is contained in this system variable will be executed whenever Natural would otherwise display the command input prompt (NEXT prompt or Direct Command line/window).

*STARTUP contains the name of the program which has been entered in Natural Security as startup transaction in the security profile of the respective library (except in batch mode; see also the Natural Security documentation).

If no startup transaction is specified, or if Natural Security is not used, *STARTUP contains the value of the profile parameter STARTUP (except on mainframe computers).

On mainframe computers, if no startup transaction is specified or if Natural Security is not used, the value of *STARTUP depends on the setting of the profile parameter MENU:

- If MENU=OFF is set, *STARTUP will be blank.
- If MENU=ON is set, *STARTUP contains "MAINMENU"; that is, the Natural main menu will be displayed.

Via a Natural program you can assign to *STARTUP a program name which will always overwrite its previous content.

Note:
A startup program used in batch mode must contain a FETCH or STACK COMMAND statement; otherwise error NAT9969 may occur.

If you invoke the command input prompt by entering "%%" (or any equivalent command) - either in a non-SECURITY environment or in a SECURITY environment in which command mode is not prohibited for the current library - the startup mechanism will be deactivated. To subsequently re-activate it, log on to the library again or execute a program which re-assigns the name of a program to *STARTUP.
In a Natural Security environment in which command mode is prohibited for the current library, "%%" will cause the program whose name is contained in *STARTUP to be invoked.

 Copyright © Software AG 2002

*STEPLIB

Format/length:	A8
Content modifiable:	No

This system variable contains the name of the Steplib library which has been concatenated to the Natural library to which the user is currently logged on.

If Natural Security is not active, *STEPLIB contains:

- under OpenVMS, UNIX and Windows: the *STEPLIB name specified with the profile parameter LSTEP in the parameter file used;
- on mainframe computers: the name specified with the profile parameter STEPLIB.

If Natural Security is active, the value may be defined in the security profile of a given library.

Note:
The database ID and file number of the *STEPLIB library are derived from its name. Apart from the library SYSTEM, libraries with the name SYSxxx are assumed to be in FNAT and other libraries are assuumed to be in FUSER.

*SUBROUTINE

Format/length:	A32
Content modifiable:	No

This system variable contains the name of the external subroutine that is currently being executed.

*THIS-OBJECT

Format/length:	HANDLE OF OBJECT
Content modifiable:	No

This system variable is only available with NaturalX. It is a handle to the currently active object. The currently active object uses *THIS-OBJECT to either execute its own methods or pass a reference to itself to another object.

*THIS-OBJECT only contains an actual value when a method is being executed. Otherwise it contains "NULL-HANDLE".

For further information, see the NaturalX documentation.

*TPSYS

Format/length: A8

Content modifiable: No

This system variable contains the name of the TP monitor that is being used.

It can contain one of the following values:

AIM/DC
CICS
COMPLETE
IMS/DC
OS/400
SERVSTUB (NDV server)
TIAM
TSO
TSS
UTM
VM/CMS
VTAM

In batch mode, *TPSYS will be blank.

If no TP monitor is used, *TPSYS contains the value "NONE".

*UI

Format/length: A16

Content modifiable: No

This system variable indicates the type of user interface being used:

Character	Character-oriented user interface.
GUI	Graphical user interface.

 Copyright © Software AG 2002

*USER

Format/length: A8

Content modifiable: No

This system variable contains the user ID as taken from the Natural Security logon.

If the profile parameter AUTO=ON (Automatic Logon) is set or if Natural Security is not active, the value of *USER is identical to that of *INIT-USER.

*USER-NAME

Format/length: A32

Content modifiable: No

If Natural Security is installed, this variable contains the name of the user who is currently logged on to Natural.

If Natural Security is not active, the default is "SYSTEM".

*WINDOW-LS

Format/length: N3

Content modifiable: No

This system variable contains the line size of the logical window (without frame). See also the DEFINE WINDOW statement.

*WINDOW-POS

Format/length: N6

Content modifiable: No

This system variable contains the position which corresponds to the upper left corner of the window. See also the DEFINE WINDOW statement.

The position is counted in characters across multiple lines, beginning with "0" (upper left corner).

*WINDOW-PS

Format/length: N3

Content modifiable: No

This system variable contains the page size of the logical window (without frame). See also the DEFINE WINDOW statement.

Copyright © Software AG 2002

*WINMGR

Format/length:	A16
Content modifiable:	No

If a graphical user interface is used, this system variable contains the name of the window manager being used (for example, MOTIF or PM).

If a character-oriented user interface is used, *WINMGR will be blank.

The type of user interface is indicated by the value of the system variable *UI.

*WINMGRVERS

Format/length:	A16
Content modifiable:	No

If a graphical user interface is used, this system variable contains the version number of the window manager being used.

If a character-oriented user interface is used, *WINMGRVERS will be blank.

The type of user interface is indicated by the value of the system variable *UI.

 Copyright © Software AG 2002

Date and Time System Variables

The date and time system variables listed below may be specified in a COMPUTE, DISPLAY, MOVE, PRINT, WRITE statement and in logical condition criteria.

The contents of date and time system variables as generated by Natural are non-modifiable, which means that in a Natural program you cannot assign another value to any of them.

All date variables contain the current date. The format of the date is different for each date variable, as indicated below.

Date Variable	Format/Length	Date Format*
***DATD**	A8	DD.MM.YY
***DAT4D**	A10	DD.MM.YYYY
***DATE**	A8	DD/MM/YY
***DAT4E**	A10	DD/MM/YYYY
***DATG**	A15	DDmonthnameYYYY (Gregorian date)
***DATI**	A8	YY-MM-DD
***DAT4I**	A10	YYYY-MM-DD
***DATJ**	A5	YYDDD (Julian date)
***DAT4J**	A7	YYYYDDD (Julian date)
***DATN**	N8	YYYYMMDD
***DATU**	A8	MM/DD/YY
***DAT4U**	A10	MM/DD/YYYY
***DATV**	A11	DD-MON-YYYY (**)
***DATVS**	A9	DDMONYYYY (**)
***DATX**	D	internal date format

* D = day, M = month, Y = year.

** The system variables *DATV and *DATVS are not available on mainframe computers.

Time Variable	Format/Length	Explanation
TIMD(r)*	N7	Can only be used in conjunction with a previous SETTIME statement. Contains the time that has elapsed after the SETTIME statement was executed (in format HHIISST (*)). (r) represents the statement label or source-code line number of the SETTIME statement used as the basis for *TIMD.
***TIME**	A10	Contains the time of day in format HH:II:SS.T (*).
***TIME-OUT**	N5	Contains the number of seconds remaining before the current transaction will be timed out (only available with Natural Security). *TIME-OUT is "0" if transaction mode has not been entered. Transaction mode is entered with the execution of a FIND, READ or GET statement that reads a database record for the purpose of updating or deleting the record. *TIME-OUT is reset to "0" when an END TRANSACTION or BACKOUT TRANSACTION statement is executed.
***TIMESTMP**	B8	Machine-internal store clock value.
***TIMN**	N7	Contains the time of day in format HHIISST (*).
***TIMX**	T	Contains the time of day in internal time format.

* H = hour, I = minute, S = second, T = tenth of a second.

Example of Date and Time System Variables:

```
DATE IN FORMAT DD.MM.YYYY   14.01.1999
DATE IN FORMAT DD/MM/YYYY   14/01/1999DATE IN GREGORIAN FORM    20January 1999DATE IN FORMAT YYYY-MM-DD  1999-01-14DATE IN FORMAT YYYYDDD    1999014DATE IN FORMAT YYYYMMDD    19990114DATE IN FORMAT MM/DD/YYYY  01/20/2000DATE IN INTERNAL FORMAT    1999-01-INTIME IN FORMAT HH:II:SS.T  15:34:04.4TIME IN FORMAT HHIISST    153404TIME IN INTERNAL FORMAT    15:34:04CURRENT DATE 1999-01-18  CURRENT DATE + 14 DAYS   1999-02 02CURRENT TIME 15:34:44    CURRENT TIME + 10 SECONDS 15:34:54
```

 Copyright © Software AG 2002

Large and Dynamic Variables/Fields

The following topics are covered below:

- Introduction
- Definition of Dynamic Variables
- System Variable *LENGTH(field)
- Size Limitation Checks
- Statements EXPAND and REDUCE
- Usage of Dynamic Variables

Introduction

As of Natural Version 4.1, enhanced capabilities are provided for the usage of large variables by removing the existing size limitations and by providing for dynamic allocation of these variables at execution time.

Large variables for alphanumeric and binary data are based on the well known Natural formats A and B. The limitations of 253 for Format A and 126 for Format B are no longer in effect. The new size limit is 1 GB. These large static variables and fields are handled in the same manner as traditional alphanumeric and binary variables and fields with regard to definition, redefinition, value space allocation, conversions, referencing in statements, etc. All rules concerning alphanumeric and binary formats apply to these large formats.

In that the maximum size of large data structures (for example, pictures, sounds, videos) may not exactly be known at application development time, Natural additionally provides for the definition of alphanumeric and binary variables with the attribute DYNAMIC. The value space of variables which are defined with this attribute will be extended dynamically at execution time when it becomes necessary (for example, during an assignment operation: #picture1 := #picture2). This means that large binary and alphanumeric data structures may be processed in Natural without the need to define a limit at development time. The execution-time allocation of dynamic variables is of course subject to available memory restrictions. If the allocation of dynamic variables results in an insufficent memory condition being returned by the underlying operating system, the ON ERROR statement can be used to intercept this error condition; otherwise, an error message will be returned by Natural. The new Natural system variable *LENGTH can be used to obtain the number of bytes of the value space which are currently used for a given dynamic variable. Natural automatically sets *LENGTH to the length of the source operand during assignments in which the dynamic variable is involved. *LENGTH(field) therefore returns the size currently used for a dynamic Natural field or variable in bytes.

If the dynamic variable space is no longer needed, the REDUCE DYNAMIC VARIABLE statement can be used to reduce the space used for the dynamic variable to zero (or any other desired size). If the upper limit of memory usage is known for a specific dynamic variable, the EXPAND statement can be used to set the space used for the dynamic variable to this specific size.

If a dynamic variable is to be initilialized, the MOVE ALL UNTIL statement should be used for this purpose.

Note concerning Natural RPC:
Large and dynamic variables are currently not supported by the Natural remote procedure call (RPC).

Definition of Dynamic Variables

Because the actual size of large alphanumeric and binary data structures may not be exactly known at application development time, the definition of ***dynamic*** variables of Format A or B can be used to manage these structures. The dynamic allocation and extension (reallocation) of large variables is transparent to the application programming logic. Dynamic variables are defined without any length. Memory allocation will be done at execution time implicitly, when the dynamic variable is used as a target operand or explicitly with an EXPAND

statement.

Dynamic variables can only be defined in a DEFINE DATA statement using the following syntax:

```
level   variable-name   ( A )    DYNAMIC
level   variable-name   ( B )    DYNAMIC
```

A dynamic variable can only be defined as **scalar** (no dynamic array definition is possible).
A dynamic variable may not be contained in a **REDEFINE clause**, and a **redefinition** of a dynamic variable or of a group that contains a dynamic variable is not possible. The **CONST** and **INIT** clauses are invalid for dynamic variables.

System Variable *LENGTH(field)

The size of the currently used value space of a dynamic variable can be obtained from the system variable *LENGTH. *LENGTH is set to the (used) length of the source operand during assignments automatically.

> **Note:**
> Due to performance considerations, the allocated size may be larger than the used size. It is not possible for the Natural programmer to obtain information about the currently allocated size. This is an internal value.

*LENGTH(field) returns the used size of a dynamic Natural field or variable in bytes. *LENGTH may be used only to get the currently used size for dynamic variables.

Size Limitation Checks

Profile Parameter DSLM

For compatibility reasons, a size limitation check at compile time for fixed length variables can be done using the DSLM parameter. The DSLM parameter limits the size to 32 KB per variable.

Profile Parameter USIZE

For dynamic variables, a size limitation check at compile time is not possible because no length is defined for dynamic variables. The Size of User Buffer Area (USIZE) indicates the size of the user buffer in virtual memory. The user buffer contains all data dynamically allocated by Natural. If a dynamic variable is allocated or extended at execution time and the USIZE limitation is exceeded, an error message will be returned.

Statements EXPAND and REDUCE

The statements EXPAND and REDUCE are used to explicitly allocate and free memory space for a dynamic variable.

Syntax:

```
EXPAND [ SIZE OF ] DYNAMIC [ VARIABLE ]  operand1  TO operand2
REDUCE [ SIZE OF ] DYNAMIC [ VARIABLE ]  operand1  TO operand2
```

where operand1 is a dynamic variable and operand2 is a non-negative numeric size value.

The EXPAND statement is used to extend the allocated size of the dynamic variable to a given size. The size currently used (*LENGTH) for the dynamic variable is not modified.
If the given size is less than the currently allocated size of the dynamic variable, the statement will be ignored.

 Copyright © Software AG 2002

The REDUCE statement is used to reduce the allocated memory size. The allocated memory of the dynamic variable beyond the given size is released immediately (when the statement is executed).
If the size currently used (*LENGTH) for the dynamic variable is greater than the given size, *LENGTH of this dynamic variable is set to this size. The content of the variable is truncated, but not modified. If the given size is larger than the currently allocated size of the dynamic variable, the statement will be ignored.

Usage of Dynamic Variables

- Assignments with Dynamic Variables
- Initialization of Dynamic Variables
- String Manipulation with Dynamic Alpha Variables
- Logical Condition Criterion (LCC) with Dynamic Variables
- Parameter Transfer with Dynamic Variables
- Work File Access with Large and Dynamic Variables
- DDM Generation and Editing for Varying Length Columns
- Accessing Large Database Objects
- Performance Aspects with Dynamic Variables

Generally, a dynamic alphanumeric variable may be used wherever an operand of Format A or Format B is allowed.

Exception:
Dynamic variables are not allowed within DISPLAY, WRITE, PRINT, STACK, INPUT, REINPUT, and SORT statements. (To DISPLAY, WRITE or PRINT dynamic alphanumeric variables, the variable must be cut into smaller portions using the MOVE SUBSTRING statement.)

The used length (*LENGTH) and the allocated size of dynamic variables are equal to zero until such time that the variable is first accessed as a target operand. Due to assignments or other manipulation operations, dynamic variables may be firstly allocated or extended (reallocated) to the exact size of the source operand.

The size of a dynamic variable may be extended if it is used as a modifiable operand (target operand) in the following statements:

- destination operand in an assignment (ASSIGN, MOVE)
- operand2 in COMPRESS
- operand1 in EXAMINE REPLACE
- operand4 in SEPARATE
- READ WORK FILE
- parameter or view field in the INTO clause of SELECT
- CALLNAT, PERFORM (except AD=O, or BY VALUE in PDA)
- SEND METHOD

Currently, there is the following limit concerning the usage of large variables:

- CALL statement parameter size less than 64 KB per parameter (no limit for the CALL with INTERFACE4 option).

In the following sections, the use of dynamic variables is discussed in more detail with examples.

Assignments with Dynamic Variables

Generally, an assignment is done in the current used length (*LENGTH) of the source operand.
If the destination operand is a dynamic variable, its current allocated size is possibly extended in order to move the source operand without truncation.

Example:

```
#MyDynText1 := operand or
MOVE operand to #MyDynText1
#MyDynText1 is automatically extended until the source operand can be copied
```

MOVE ALL, MOVE ALL UNTIL with dynamic target operands are defined as follows:

- `MOVE ALL` moves the source operand repeatedly to the target operand until the used length (*LENGTH) of the target operand is reached. *LENGTH is not modified. If *LENGTH is zero, the statement will be ignored.
- `MOVE ALL operand1 TO operand2 UNTIL operand3` moves operand1 repeatedly to operand2 until the length specified in operand3 is reached. If operand3 is greater than *LENGTH(operand2), operand2 is extended and *LENGTH(operand2) is set to operand3. If operand3 is less than *LENGTH(operand2), the used length is reduced to operand3. If operand 3 equals *LENGTH(operand2), the behavior is equivalent to MOVE ALL.

Example:

```
#MyDynText1 := 'ABCDEFGHIJKLMNO'          /* *LENGTH(#MyDynText1) is 15
MOVE ALL 'AB' TO #MyDynText1              /* content of #MyDynText1 is 'ABABABABABABABA';
                                          /* *LENGTH is still 15
MOVE ALL 'CD' TO #MyDynText1 UNTIL 6      /* content of #MyDynText1 is 'CDCDCD';
                                          /* *LENGTH is 6
MOVE ALL 'EF' TO #MyDynText1 UNTIL 10     /* content of #MyDynText1 is 'EFEFEFEFEF';
                                          /* *LENGTH is 10
```

MOVE JUSTIFIED is rejected at compile time if the target operand is a dynamic variable.

MOVE SUBSTR and MOVE TO SUBSTR are allowed. MOVE SUBSTR will lead to a runtime error if a sub-string behind the used length of a dynamic variable (*LENGTH) is referenced. MOVE TO SUBSTR will lead to a runtime error if a sub-string position behind *LENGTH + 1 is referenced, because this would lead to an undefined gap in the content of the dynamic variable. If the target operand should be extended by MOVE TO SUBSTR (for example if the second operand is set to *LENGTH+1), the third operand is mandatory.

Example:

```
/* valid
#op2 := *LENGTH(#MyDynText1)
MOVE SUBSTR (#MyDynText1, #op2) TO operand                /* move last character to operand
#op2 := *LENGTH(#MyDynText1) + 1
MOVE operand TO SUBSTR(#MyDynText1, #op2, #len_operand)
                                                          /* concatenate operand to #MyDynText1
/* invalid
#op2 := *LENGTH(#MyDynText1) +1
MOVE SUBSTR (#MyDynText1, #op2, 10) TO operand            /* leads to runtime error; undefined sub-string
#op2 := *LENGTH(#MyDynText1 +10)
MOVE operand TO SUBSTR(#MyDynText1, #op2, #len_operand)
       /* leads to runtime error; undefined gap
#op2 := *LENGTH(#MyDynText1) +1
MOVE operand TO SUBSTR(#MyDynText1, #op2)                 /* leads to runtime error; undefined length
```

Assignment Compatibility

Example:

```
#MyDynText1   := #MyStaticVar1
#MyStaticVar1 := #MyDynText2
```

 Copyright © Software AG 2002

If the source operand is a static variable, the used length of the dynamic destination operand (*LENGTH(#MyDynText1)) is set to the format length of the static variable and the source operand is copied in this length including trailing blanks (Format A) or zeros (Format B).

If the destination operand is static and the source operand is dynamic, the dynamic variable is copied in its currently used size. If this size is less than the format length of the static variable, the rest is filled with blanks or zeros. Otherwise, the value will be truncated. If the currently used size of the dynamic variable is 0, the static target operand is filled with blanks or zeros.

Initialization of Dynamic Variables

Dynamic Variables can be initialized with blanks (alphanumeric) or zeros (binary) up to the currently used length (= *LENGTH) using the RESET statement. *LENGTH is not modified.

Example:

```
DEFINE DATA LOCAL
:
#MyDynText1   (A)  DYNAMIC
:
END-DEFINE
:
#MyDynText1 := 'short text'
write *LENGTH(#MyDynText1)          /* used length is 10
:
RESET #MyDynText1                   /* used length is still 10, value is 10 blanks
:
```

To initialize a dynamic variable with a specified value in a specified size, the MOVE ALL UNTIL statement may be used.

Example:

```
:
MOVE ALL 'Y' TO #MyDynText1 UNTIL 15         /* #MyDynText1 contains 15 'Y's, used length is 15
:
```

String Manipulation with Dynamic Alpha Variables

If a modifiable operand is a dynamic variable, its current allocated size is possibly extended in order to perform the operation without truncation or an error message. This is valid for the concatenation (COMPRESS) and separation of dynamic alphanumeric variables (SEPARATE).

Example:

```
DEFINE DATA LOCAL
   1  #MyDynText1          (A)  DYNAMIC
   1  #MyDynText2          (A)  DYNAMIC
...
COMPRESS ... INTO #MyDynText2

#MyDynText2 will be extended in order to compress the source operands.
Note: in case of non-dynamic variables the value may be truncated.

SEPARATE INTO #MyDynText1 #MyDynText2 WITH DELIMITER
#MyDynText1 and #MyDynText2 are possibly extended or reduced to separate the source operand.

EXAMINE #MyDynText1 FOR REPLACE
#MyDyntext1 will possibly be extended or reduced to perform the REPLACE operation successfully.
```

Note: In case of non-dynamic variables, an error message may be returned.

 Copyright © Software AG 2002

Logical Condition Criterion (LCC) with Dynamic Variables

Generally, a read-only operation (such as LCC) with a dynamic variable is done with its currently used size. Dynamic variables are processed like static variables if they are used in a read-only (non-modifiable) context.

Example:

```
IF  #MyDynText1  =  #MyDynText2  OR  #MyDynText1 =  "**"
IF  #MyDynText1  <  #MyDynText2  OR  #MyDynText1 <  "**"
IF  #MyDynText1  >  #MyDynText2  OR  #MyDynText1 >  "**"
```

Also in the case of trailing blanks or zeros, dynamic variables will show an equivalent behavior.
For dynamic variables, the alphanumeric value ’AA ’ will be equal to ’AA’ and the binary value ’00003031’ is equal to ’3031’. If a comparison result should only be TRUE in case of an exact copy, the used lengths of the dynamic variables have to be compared in addition. For example, a variable for a small music clip and a variable for the same clip with a following period of silence at the end could be compared. If one variable is an exact copy of the other, their used lengths are also equal.

Example:

```
#MyDynText1  :=  ’hello’          /* used length is 5
#MyDynText2  :=  ’hello     ’     /* used length is 10
IF  #MyDynText1  =  #MyDynText2   /* TRUE
:
IF   #MyDynText1  =  #MyDynText2
  AND  *LENGTH(#MyDynText1) = *LENGTH(#MyDynText2)
                                  /* FALSE
:
```

Two dynamic variables are compared position by position from left to right up to the minimum of their used lengths. The first position where the variables are not equal determines if the first or the second variable is greater than, less than or equal to the other. The variables are equal if they are equal up to the minimum of their used lengths and the rest of the longer variable contains only blanks (Format A) or zeros (Format B).

Example:

```
#MyDynText1  :=  ’hello1’                /* used length is 6
#MyDynText2  :=  ’hello2’                /* used length is 10
IF  #MyDynText1   #MyDynText2            /* TRUE : #MyDynText2
:="hallo" IF #MyDynText1>  #MyDynText2   /* TRUE
:
```

Comparison Compatibility

Comparisons between dynamic and static variables are equivalent to comparisons between dynamic variables. The format length of the static variable is interpreted as its used length.

Example:

```
#MyStatText1 :=  'hello'            /* format length of MyStatText1 is 20
#MyDynText1  :=  'hello'            /* used length is 5
IF  #MyStatText1  =  #MyDynText1    /* TRUE
:
IF  #MyStatText1  >  #MyDynText1    /* FALSE
```

 Copyright © Software AG 2002

Parameter Transfer with Dynamic Variables

Dynamic variables may be passed as parameters to a called program object (CALLNAT, PERFORM).
Call-by-reference is possible because the value space of a dynamic variable is contiguous. Call-by-value causes an assignment with the variable definition of the caller as the source operand and the parameter definition as the destination operand. Call-by-value result causes in addition the movement in the opposite direction.
For call-by-reference, both definitions must be DYNAMIC. If only one of them is DYNAMIC, a runtime error is raised. In the case of call-by-value (result), all combinations are possible. The following table illustrates the valid combinations:

Call By Reference

	Parameter	
Caller	**Static**	**Dynamic**
Static	Yes	No
Dynamic	No	Yes

The formats of dynamic variables A or B must match.

Call by Value (Result)

	Parameter	
Caller	**Static**	**Dynamic**
Static	Yes	Yes
Dynamic	Yes	Yes

Note:
In the case of static/dynamic or dynamic/static definitions, a value truncation may occur according to the data transfer rules of the appropriate assignments.

Copyright © Software AG 2002

Example 1:

```
DEFINE DATA LOCAL
    1 #MyText (A) DYNAMIC
:
#MyText :=  '123456'            /* extended to 6 bytes
WRITE *LENGTH(#MyText)         /* is 6
CALLNAT  'SUB1' USING #MyText
WRITE *LENGTH(#MyText)         /* is 8; allocated size is 8


Subpgm SUB1:
DEFINE DATA PARAMETER
    1 #MyParm (A)  DYNAMIC BY VALUE RESULT
:
WRITE *LENGTH(#MyParm)     /*is 6; temporary value space of 6 bytes is allocated for #MyParm
#MyParm := '1234567'       /* extended to 7
#MyParm := '12345678'      /* allocated size=8 bytes
EXPAND DYNAMIC VARIABLE #MyParam TO 10
WRITE *LENGTH(#MyParm)     /* is 8; allocated size is 10
END                        /* contents of  #Myparm is moved (back) to #MyText
                           /* used length is 8; #MyText is extended to 8
```

 Copyright © Software AG 2002

Example 2:

```
DEFINE DATA LOCAL
    1 #MyText (A) DYNAMIC
:
#MyText :=  '123456'                  /* extended to 6 bytes
WRITE *LENGTH(#MyText)                /* is 6
CALLNAT  'SUB2' USING #MyText
WRITE *LENGTH(#MyText)                /* is 8; allocated size is 10 (extended in SUB2)


Subpgm SUB2:
DEFINE DATA PARAMETER
    1 #MyParm (A)  DYNAMIC
:
WRITE *LENGTH(#MyParm)                /* is 6
#MyParm := '1234567'                  /* used length is 7
#MyParm := '12345678'                 /* extended to 8 bytes
EXPAND DYNAMIC VARIABLE #MyParm TO 10
WRITE *LENGTH(#MyParm)                /* is 8; allocated size is 10
END
```

CALL 3GL Program

Dynamic and large variables can sensibly be used with the CALL statement when the option INTERFACE4 is used. The usage of this option leads to an interface to the 3GL program with a different parameter structure. This usage requires some minor changes in the 3GL program, but provides the following significant benefits as compared with the older FINFO structure. For further information on the FINFO structure, see the Call Interface4 statement.

- No limitation on the number of passed parameters (former limit 40).
- No limitation on the parameter's data size (former limit 64 KB per parameter).
- Full parameter information can be passed to the 3GL program including array information.
  Exported functions are provided which allow secure access to the parameter data (formerly you had to take care not to overwrite memory inside of Natural)

Before calling a 3GL program with dynamic parameters, it is important to ensure that the necessary buffer size is allocated. This can be done explicitly with the EXPAND statement.
If an initialized buffer is required, the dynamic variable can be set to the initial value and to the necessary size by using the MOVE ALL UNTIL statement. Natural provides a set of functions that allow the 3GL program to obtain information about the dynamic parameter and to modify the length when parameter data is passed back.

Example:

```
MOVE ALL ' ' TO  #MyDynText1 UNTIL 10000      /* a buffer of length 10000 is allocated
                                              /* #MyDynText1 is initialized with blanks
                                              /* *LENGTH is set is set to 10000
CALL INTERFACE4  #3GLprog  USING  #MyDynText1
write *LENGTH(#MyDynText1)                    /* used length may have changed in the 3GL program
:
```

For a more detailed description, refer to the CALL statement in the Statements documentation.

 Copyright © Software AG 2002

Work File Access with Large and Dynamic Variables

Large and Dynamic Variables can be written into work files or read from work files using the two work file types PORTABLE and UNFORMATTED. For these types, there are no size restrictions for large/dynamic variables.

The other work file types (ASCII, ASCII-COMPRESSED, ENTIRECONNECTION, SAG and TRANSFER) cannot handle dynamic variables and will produce an error. Large variables for these work file types pose no problem unless the maximum field/record length is exceeded (Field Length 255 for ENTIRECONNECTION and TRANSFER, Record Length 32767 for the others).

For the work file type PORTABLE, the field information is stored within the work file. The dynamic variables are resized during READ if the field size in the record is different from the current size.

The work file type UNFORMATTED can be used, for example, to read a video from a database and store it in a file directly playable by other utilities. In the WRITE WORK statement, the fields are written to the file with their byte length. All data types (DYNAMIC or not) are treated the same. No structural information is inserted. Note that Natural uses a buffering mechanism, so you can expect the data to be completely written only after a CLOSE WORK. This is especially important if the file is to be processed with another utility while Natural is running.

With the READ WORK statement, fields of fixed length are read with their whole length. If the end-of-file is reached, the rest of the current field is filled with blanks. The following fields are unchanged.
In the case of DYNAMIC data types, all the rest of the file is read unless it exceeds 1 GB. If the end of file is reached, the remaining fields (variables) are kept unchanged (normal Natural behavior).

DDM Generation and Editing for Varying Length Columns

Depending on the data types, the related database format A or format B is generated. For the databases' data type VARCHAR the NATURAL length of the column is set to the maximum length of the data type as defined in the DBMS. In case of large data types the keyword DYNAMIC is generated at the length field position.

For all varying length columns, an LINDICATOR field L@<column-name> will be generated. For the databases' data type VARCHAR, an LINDICATOR field with Format/Length I2 will be generated. For large data types (see list below) the format/length will be I4.

In the context of database access, the LINDICATOR handling offers the chance to get the size of the field to be read or to set the size of the field to be written independent of a defined buffer length (or independent of *LENGTH). Usually, after a retrieval function, *LENGTH will be set to the corresponding length indicator value.

Copyright © Software AG 2002

Example DDM:

```
 T  L  Name                     F   Leng        S   D   Remark
    :
    1  L@ PICTURE1               I   4                                   /* length indicator
    1  PICTURE1                  B   DYNAMIC             IMAGE
    1  N@PICTURE1                I   2                                   /* NULL indicator
    1  L@TEXT1                   I   4                                   /* length indicator
    1  TEXT1                     A   DYNAMIC             TEXT
    1  N@TEXT1                   I   2                                   /* NULL indicator
    1  L@DESCRIPTION             I   2                                   /* length indicator
    1  DESCRIPTION               A   1000                VARCHAR(1000)
    :
    :
~~~~~~~~~~~~~~~~~~~~~~~Extended Attributes~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~/* concerning PICTURE1
 Header               :    ---
 Edit Mask            :    ---
 Remarks              :   IMAGE
```

The generated formats are varying length formats. The Natural programmer has the chance to change the definition from DYNAMIC to a fixed length definition (extended field editing) and can change, for example, the corresponding DDM field definition for VARCHAR data types to a multiple value field (old generation).

Example:

```
 T  L  Name                 F   Leng          S   D   Remark
    :
    1  L@ PICTURE1           I   4                                   /* length indicator
    1  PICTURE1              B   1000000000            IMAGE
    1  N@PICTURE1            I   2                                   /* NULL indicator
    1  L@TEXT1               I   4                                   /* length indicator
    1  TEXT1                 A   5000                  TEXT
    1  N@TEXT1               I   2                                   /* NULL indicator
    1  L@DESCRIPTION         I   2                                   /* length indicator
M   1  DESCRIPTION           A   100                   VARCHAR(1000)
      :
      :
~~~~~~~~~~~~~~~~~~~~~Extended Attributes~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~/* concerning PICTURE1
 Header             :    ---
 Edit Mask          :    ---
 Remarks            :   IMAGE
```

Accessing Large Database Objects

To access a database with large objects (CLOBs or BLOBs), a DDM with corresponding large alphanumeric or binary fields is required. If a fixed length is defined and if the database large object does not fit into this field, the large object is truncated. If the programmer does not know the definitive length of the database object, it will make sense to work with dynamic fields. As many reallocations as necessary are done to hold the object. No truncation is performed.

 Copyright © Software AG 2002

Example Program:

```
DEFINE DATA LOCAL
:
1 person VIEW OF xyz-person
  2 nachname
  2 vorname_1
  2 L@PICTURE1                  /* I4 length indicator for PICTURE1
  2 PICTURE1                    /* defined as dynamic in the DDM
  2 TEXT1                       /* defined as non-dynamic in the DDM
:
END-DEFINE
:
SELECT * INTO VIEW person FROM xyz-person                  /* PICTURE1 will be read completely
                            WHERE nachname = 'SMITH'       /* TEXT1 will be truncated to fixed length 5000
:
   WRITE  'length of PICTURE1: '  L@PICTURE1               /* the L-INDICATOR will contain the length
   :                                                       /* of PICTURE1 (= *LENGTH(PICTURE1)
   /* do something with PICTURE1 and TEXT1
:
   L@PICTURE1  :=  100000
   INSERT INTO xyz-person  (*) VALUES (VIEW person)        /* only the first 100000 Bytes of PICTURE1
   :                                                       /* are inserted
:
END-SELECT
```

If a format-length definition is omitted in the view, this is taken from the DDM.
In reporting mode, it is now possible to specify any length, if the corresponding DDM field is defined as DYNAMIC. The dynamic field will be mapped to a field with a fixed buffer length. The other way round is not possible.

DDM format/length definition	**VIEW format / length definition**	
(An)	-	valid
	(An)	valid
	(Am)	only valid in reporting mode
	(A) DYNAMIC	invalid
(A) DYNAMIC	-	valid
	(A) DYNAMIC	valid
	(An)	only valid in reporting mode
	(Am / i : j)	only valid in reporting mode

(equivalent for Format B variables)

Parameter with LINDICATOR Clause in SQL Statements

If the LINDICATOR field is defined as I2 field, the SQL data type VARCHAR is used for sending or receiving the corresponding column. If the LINDICATOR host variable is specified as I4, a large object data type (CLOB/BLOB) is used.

If the field is defined as DYNAMIC, the column is read in an internal loop up to its real length. The LINDICATOR field and *LENGTH are set to this length. In the case of a fixed-length field, the column is read up to the defined length. In both cases, the field is written up to the value defined in the LINDICATOR field.

Performance Aspects with Dynamic Variables

EXPAND and REDUCE

The amount of the allocated memory of a dynamic variable may be reduced to a specified size using the REDUCE DYNAMIC VARIABLE statement. In order to (re)allocate a variable to a specified size, the EXPAND statement can be used. (If the variable should be initialized, use the MOVE ALL UNTIL statement.)

Example:

```
DEFINE DATA LOCAL
:
#MyDynText1    (A)  DYNAMIC
# len          (I4)
:
END-DEFINE

#MyDynText1 := 'a'                  /* used length is 1, value is 'a'; allocated size is still 1

EXPAND DYNAMIC VARIABLE  #MyDynText1 TO 100
                                    /* used length is still 1, value is 'a'; allocated size is 100

CALLNAT #subprog  USING  #MyDynText1
write *LENGTH(#MyDynText1)          /* used length and allocated size may have changed in the subprogram

#len := *LENGTH(#MyDynText1)
REDUCE DYNAMIC VARIABLE  #MyDynText1  TO  #len
                                    /* if allocated size is greater than used length, the unused memory is released
:
REDUCE DYNAMIC VARIABLE  #MyDynText1  TO  0
                                    /* free allocated memory for dynamic variable
END
```

Rules:

- Use dynamic operands where it makes sense.
- Use EXPAND if upper limit of memory usage is known.
- Use REDUCE if the dynamic operand will no longer be needed.

 Copyright © Software AG 2002

Natural System Functions

This document describes various Natural "built-in" functions for use in certain statements.

The following topics are covered:

- Using Natural System Functions
- Alphabetical List of Natural System Functions
- Mathematical Functions
- POS - Field Identification Function
- RET - Return Code Function
- SORTKEY - Sort-Key Function

Using Natural System Functions

The Natural system functions listed below may be specified in a MOVE, COMPUTE, DISPLAY, PRINT or WRITE statement that is used within any of the following statement blocks:

- AT BREAK,
- AT END OF DATA,
- AT END OF PAGE,

that is, for all FIND, READ, HISTOGRAM, SORT or READ WORK FILE processing loops.

If a system function is used within an AT END OF PAGE statement, the corresponding DISPLAY statement must include the GIVE SYSTEM FUNCTIONS clause.

Records rejected by a WHERE clause are not evaluated by a system function.

If system functions are evaluated from database fields which originated from different levels of processing loops initiated with a FIND, READ, HISTOGRAM or SORT statement, the values are always processed according to their position in the loop hierarchy. For example, values for an outer loop will only be processed when new data values have been obtained for that loop.

If system functions are evaluated from user-defined variables, the processing is dependent on the position in the loop hierarchy where the user-defined variable was introduced in reporting mode. If the user-defined variable is defined before any processing loop is initiated, it will be evaluated for system functions in the loop where the AT BREAK, AT END OF DATA or AT END OF PAGE statement is defined. If a user-defined variable is introduced within a processing loop it will be processed the same as a database field from that processing.

For selective referencing of system function evaluation for user-defined variables it is recommended to specify a loop reference with the user-defined variable to indicate in which loop the value is to be processed. The loop reference may be specified as a statement label or source code line number.

Copyright © Software AG 2002

System Functions in SORT GIVE FUNCTIONS Statement

System functions may also be referenced when they have been evaluated in a GIVE FUNCTIONS clause of a SORT statement.

For a reference to a system function evaluated with a SORT GIVE FUNCTIONS statement, the name of the system function must be prefixed with an asterisk (*).

Arithmetic Overflows in AVER, NAVER, SUM or TOTAL

Fields to which the system functions AVER, NAVER, SUM and TOTAL are to be applied must be long enough (either by default or user-specified) to hold any overflow digits. If any arithmetic overflow occurs, an error message will be issued.

Normally, the length is the same as that of the field to which the system function is applied; if this is not long enough, use the NL parameter to increase the output length as follows:

SUM(field)(NL=nn)

This will not only increase the output length but also causes the field to be made longer internally.

Statement Referencing *(r)*

Statement referencing is also available for system functions.

By using a statement label or the source-code line number *(r)* you can determine in which processing loop the system function is to be evaluated for the specified field.

 Copyright © Software AG 2002

Alphabetical List of Natural System Functions

- AVER(r)(field)
- COUNT(r)(field)
- MAX(r)(field)
- MIN(r)(field)
- NAVER(r)(field)
- NCOUNT(r)(field)
- NMIN(r)(field)
- OLD(r)(field)
- SUM(r)(field)
- TOTAL(r)(field)

AVER*(r)(field)*

Format/length: Same as field.
Exception: for a field of format N, AVER*(field)* will be of format P (with the same length as the field).

This system function contains the average of all values encountered for the field specified with AVER. AVER is updated when the condition under which AVER was requested is true.

COUNT*(r)(field)*

Format/length: P7

COUNT is incremented by 1 on each pass through the processing loop in which it is located. COUNT is incremented regardless of the value of the field specified with COUNT.

MAX*(r)(field)*

Format/length: Same as field.

This system function contains the maximum value encountered for the field specified with MAX. MAX is updated (if appropriate) each time the processing loop in which it is contained is executed.

MIN*(r)(field)*

Format/length: Same as field.

This system function contains the minimum value encountered for the field specified with MIN. MIN is updated (if appropriate) each time the processing loop in which it is located is executed.

NAVER*(r)(field)*

Format/length: Same as field.
Exception: for a field of format N, NAVER*(field)* will be of format P (with the same length as the field).

This system function contains the average of all values - excluding null values - encountered for the field specified with NAVER. NAVER is updated when the condition under which NAVER was requested is true.

NCOUNT*(r)(field)*

Format/length: P7

NCOUNT is incremented by 1 on each pass through the processing loop in which it is located unless the value of the field specified with NCOUNT is a null value.

NMIN*(r)(field)*

Format/length: Same as field.

This system function contains the minimum value encountered - excluding null values - for the field specified with NMIN. NMIN is updated (if appropriate) each time the processing loop in which it is located is executed.

OLD*(r)(field)*

Format/length: Same as field.

This system function contains the value which the field specified with OLD contained prior to a control break as specified in an AT BREAK condition, or prior to the end-of-page or end-of-data condition.

SUM*(r)(field)*

Format/length: Same as field.
Exception: for a field of format N, SUM*(field)* will be of format P (with the same length as the field).

This system function contains the sum of all values encountered for the field specified with SUM. SUM is updated each time the loop in which it is located is executed. When SUM is used following an AT BREAK condition, it is reset after each value break. Only values that occur between breaks are added.

TOTAL*(r)(field)*

Format/length: Same as field.
Exception: for a field of format N, TOTAL*(field)* will be of format P (with the same length as the field).

This system function contains the sum of all values encountered for the field specified with TOTAL in all open processing loops in which TOTAL is located.

System Functions Example 1:

```
/* EXAMPLE 'ATBEX3:' AT BREAK WITH NATURAL SYSTEM FUNCTIONS
  /* ****************************************************************
  DEFINE DATA LOCAL
  1 EMPLOY-VIEW VIEW OF EMPLOYEES
    2 NAME
    2 CITY
    2 SALARY (1)
    2 CURR-CODE (1)
```

 Copyright © Software AG 2002

```
END-DEFINE
/* ******************************************************************
LIMIT 3
READ EMPLOY-VIEW LOGICAL BY CITY = 'SALT LAKE CITY'
  DISPLAY NOTITLE CITY NAME 'SALARY' SALARY(1) 'CURRENCY' CURR-CODE(1)
  AT BREAK OF CITY
     WRITE /  OLD(CITY) (EM=X^X^X^X^X^X^X^X^X^X^X^X^X^X^X^X^X)
      31T '   MINIMUM:' MIN(SALARY(1))  CURR-CODE(1) /
      31T '   AVERAGE:' AVER(SALARY(1)) CURR-CODE(1) /
      31T '   MAXIMUM:' MAX(SALARY(1))  CURR-CODE(1) /
      31T '       SUM:' SUM(SALARY(1))  CURR-CODE(1) /
      35T COUNT(SALARY(1)) 'RECORDS FOUND' /
  END-BREAK
  AT END OF DATA
     WRITE 22T 'TOTAL (ALL RECORDS):'
        T*SALARY TOTAL(SALARY(1))  CURR-CODE(1)
  END-ENDDATA
END-READ
/* ******************************************************************
END
```

```
        CITY                 NAME           SALARY   CURRENCY
-------------------- -------------------- ---------- --------

SALT LAKE CITY       ANDERSON                  50000 USD
SALT LAKE CITY       SAMUELSON                 24000 USD

S A L T   L A K E   C I T Y      MINIMUM:      24000 USD
                                 AVERAGE:      37000 USD
                                 MAXIMUM:      50000 USD
                                     SUM:      74000 USD
                                         2 RECORDS FOUND

SAN DIEGO            GEE                       60000 USD

S A N   D I E G O                MINIMUM:      60000 USD
                                 AVERAGE:      60000 USD
                                 MAXIMUM:      60000 USD
                                     SUM:      60000 USD
                                         1 RECORDS FOUND

                     TOTAL (ALL RECORDS):     134000 USD
```

System Functions Example 2:

```
/* EXAMPLE 'ATBEX4': AT BREAK USING NATURAL SYSTEM FUNCTIONS
  /*************************************************************
  DEFINE DATA LOCAL
    1 EMPLOY-VIEW VIEW OF EMPLOYEES
      2 NAME
      2 CITY
      2 SALARY (2)
    1 #INC-SALARY (P11)
  END-DEFINE
  /*************************************************************
  LIMIT 4
  EMPLOOP. READ EMPLOY-VIEW BY CITY STARTING FROM 'ALBU'
            COMPUTE #INC-SALARY = SALARY (1) + SALARY (2)
            DISPLAY NAME CITY SALARY (1:2) 'CUMULATIVE' #INC-SALARY
            SKIP
            AT BREAK CITY
```

```
              WRITE NOTITLE
                'AVERAGE:' T*SALARY (1) AVER(SALARY(1)) /
                'AVERAGE CUMULATIVE:' T*#INC-SALARY
                                      AVER(EMPLOOP.) (#INC-SALARY)
          END-BREAK
         END-READ
/*************************************************************
END
```

```
        NAME                 CITY           ANNUAL    CUMULATIVE
                                            SALARY
------------------- -------------------- ---------- ------------

HAMMOND             ALBUQUERQUE               22000        42200
                                              20200

ROLLING             ALBUQUERQUE               34000        65200
                                              31200

FREEMAN             ALBUQUERQUE               34000        65200
                                              31200

LINCOLN             ALBUQUERQUE               41000        78700
                                              37700

AVERAGE:                                      32750
AVERAGE CUMULATIVE:                                        62825
```

System Functions Example 3:

```
/* EXAMPLE 'AEDEX1S': AT END OF DATA (STRUCTURED MODE)
  DEFINE DATA LOCAL
    1 EMPLOY-VIEW VIEW OF EMPLOYEES
      2 PERSONNEL-ID
      2 NAME
      2 FIRST-NAME
      2 SALARY (1)
      2 CURR-CODE (1)
  END-DEFINE
  /*
  LIMIT 5
  EMP. FIND EMPLOY-VIEW WITH CITY = 'STUTTGART'
        IF NO RECORDS FOUND
          ENTER
        END-NOREC
        DISPLAY PERSONNEL-ID NAME FIRST-NAME SALARY (1) CURR-CODE (1)
  /*********************************************************************
        AT END OF DATA
          IF *COUNTER (EMP.) = 0
             WRITE 'NO RECORDS FOUND'
             ESCAPE BOTTOM
          END-IF
          WRITE NOTITLE / 'SALARY STATISTICS:'
                        / 7X 'MAXIMUM:' MAX(SALARY(1)) CURR-CODE (1)
                        / 7X 'MINIMUM:' MIN(SALARY(1)) CURR-CODE (1)
                        / 7X 'AVERAGE:' AVER(SALARY(1)) CURR-CODE (1)
        END-ENDDATA
  /*********************************************************************
  END-FIND
  END
```

 Copyright © Software AG 2002

```
PERSONNEL         NAME               FIRST-NAME        ANNUAL   CURRENCY
   ID                                                   SALARY     CODE
--------- -------------------- -------------------- ---------- --------

11100328  BERGHAUS             ROSE                      70800
11100329  BARTHEL              PETER                     42000
11300313  AECKERLE             SUSANNE                   55200
11300316  KANTE                GABRIELE                  61200
11500304  KLUGE                ELKE                      49200

SALARY STATISTICS:
       MAXIMUM:      70800
       MINIMUM:      42000
       AVERAGE:      55680
```

System Functions Example 4:

```
/* EXAMPLE 'AEPEX1S': AT END OF PAGE (STRUCTURED MODE)
  /*********************************************************************
  DEFINE DATA LOCAL
  1 EMPLOY-VIEW VIEW OF EMPLOYEES
    2 PERSONNEL-ID
    2 NAME
    2 JOB-TITLE
    2 SALARY (1)
    2 CURR-CODE (1)
  END-DEFINE
  /*********************************************************************
  FORMAT PS=10
  LIMIT 10
  READ EMPLOY-VIEW BY PERSONNEL-ID FROM '20017000'
    DISPLAY NOTITLE GIVE SYSTEM FUNCTIONS
            NAME JOB-TITLE 'SALARY' SALARY(1) CURR-CODE (1)
  /*********************************************************************
    AT END OF PAGE
      WRITE / 28T 'AVERAGE SALARY: ...' AVER(SALARY(1)) CURR-CODE (1)
    END-ENDPAGE
  /*********************************************************************
  END-READ
  /*********************************************************************
  END
```

```
        NAME                 CURRENT            SALARY   CURRENCY
                            POSITION                       CODE
-------------------- ------------------------- ---------- --------

CREMER               ANALYST                        34000 USD
MARKUSH              TRAINEE                        22000 USD
GEE                  MANAGER                        39500 USD
KUNEY                DBA                            40200 USD
NEEDHAM              PROGRAMMER                     32500 USD
JACKSON              PROGRAMMER                     33000 USD

                           AVERAGE SALARY: ...      33533 USD
```

Mathematical Functions

The following mathematical functions are supported in arithmetic processing statements (ADD, COMPUTE, DIVIDE, MULTIPLY, SUBTRACT) and in logical condition criteria:

Function	Format/Length	Explanation
ABS*(field)*	same as *field*	Absolute value of *field*.
ATN*(field)*	F8 (*)	Arc tangent of *field*.
COS*(field)*	F8 (*)	Cosine of *field*. If the value of the *field* is equal to or greater than 10^{17}, COS*(field)* will be "1".
EXP*(field)*	F8 (*)	Exponential of *field*.
FRAC*(field)*	same as *field*	Fractional part of *field*.
INT*(field)*	same as *field*	Integer part of *field*.
LOG*(field)*	F8 (*)	Natural logarithm of *field*.
SGN*(field)*	same as *field*	Sign of *field* (-1, 0, +1).
SIN*(field)*	F8 (*)	Sine of *field*. If the value of the *field* is equal to or greater than 10^{17}, SIN*(field)* will be "0".
SQRT*(field)*	(*) (**)	Square root of *field*. A negative value in the argument field will be treated as positive. On mainframe computers, the maximum number of digits before the decimal point of the argument is 22.
TAN*(field)*	F8 (*)	Tangent of *field*. If the value of the *field* is equal to or greater than 10^{17}, TAN*(field)* will be "0".
VAL*(field)*	same as target field	Extract numeric value from an alphanumeric *field*. The content of the *field* must be the character representation of a numeric value. Leading or trailing blanks in the *field* will be ignored; decimal point and leading sign character will be processed. If the target field is not long enough, decimal digits will be truncated (see also Field Truncation and Field Rounding).

* On all platforms, except on mainframe computers, these functions are evaluated as follows: The argument is converted to format/length F8 and then passed to the operating system for computation; the result returned by the operating system has format/length F8, which is then converted to the target format.

** On mainframe computers, the following applies:
If *field* has format/length F4, format/length of SQRT*(field)* will be F4;
if *field* has format/length F8 or I, format/length of SQRT*(field)* will be F8;
if *field* has format N or P, format/length of SQRT*(field)* will be N*n*.7 or P*n*.7 respectively (where *n* is automatically calculated to be large enough).

A *field* to be used with a mathematical function - except VAL - may be a constant or a scalar; its format must be numeric, packed numeric, integer, or floating point (N, P, I or F).

A *field* to be used with the VAL function may be a constant, a scalar, or an array; its format must be alphanumeric.

 Copyright © Software AG 2002

Mathematical Functions Example:

```
/* EXAMPLE 'MATHEX': MATHEMATICAL FUNCTIONS
  /*************************************************************
  DEFINE DATA LOCAL
  1 #A (N2.1) INIT <10>
  1 #B (N2.1) INIT <-6.3>
  1 #C (N2.1) INIT <0>
  1 #LOGA  (N2.6)
  1 #SQRTA (N2.6)
  1 #TANA  (N2.6)
  1 #ABS (N2.1)
  1 #FRAC (N2.1)
  1 #INT (N2.1)
  1 #SGN (N1)
  END-DEFINE
  /*************************************************************
  COMPUTE #LOGA  = LOG(#A)
  WRITE NOTITLE '=' #A 5X 'LOG' 40T #LOGA
  /*************************************************************
  COMPUTE #SQRTA = SQRT(#A)
  WRITE         '=' #A 5X 'SQUARE ROOT' 40T #SQRTA
  /*************************************************************
  COMPUTE #TANA  = TAN(#A)
  WRITE         '='  #A 5X 'TANGENT' 40T #TANA
  /*************************************************************
  COMPUTE #ABS   = ABS(#B)
  WRITE     //  '='  #B 5X 'ABSOLUTE' 40T #ABS
  /*************************************************************
  COMPUTE #FRAC  = FRAC(#B)
  WRITE         '='  #B 5X 'FRACTIONAL' 40T #FRAC
  /*************************************************************
  COMPUTE #INT   = INT(#B)
  WRITE         '='  #B 5X 'INTEGER' 40T #INT
  /*************************************************************
  COMPUTE #SGN   = SGN(#A)
  WRITE      // '='  #A 5X 'SIGN'     40T #SGN
  /*************************************************************
  COMPUTE #SGN   = SGN(#B)
  WRITE         '='  #B 5X 'SIGN'     40T #SGN
  /*************************************************************
  COMPUTE #SGN   = SGN(#C)
  WRITE         '='  #C 5X 'SIGN'     40T #SGN
  /*************************************************************
  END
```

```
#A:  10.0     LOG                        2.302585
#A:  10.0     SQUARE ROOT                3.162277
#A:  10.0     TANGENT                    0.648360


#B:  -6.3     ABSOLUTE                   6.3
#B:  -6.3     FRACTIONAL                -0.3
#B:  -6.3     INTEGER                   -6.0


#A:  10.0     SIGN                      1
#B:  -6.3     SIGN                     -1
#C:   0.0     SIGN                      0
```

 Copyright © Software AG 2002

POS - Field Identification Function

Format/length: I4

The system function POS*(field-name)* contains the internal identification of the field whose name is specified with the system function.

POS*(field-name)* may be used to identify a specific field, regardless of its position in a map. This means that the sequence and number of fields in a map may be changed, but POS*(field-name)* will still uniquely identify the same field. With this, for example, you need only a single REINPUT statement to make the field to be MARKed dependent on the program logic.

Example:

```
DECIDE ON FIRST VALUE OF ...
   VALUE ...
      COMPUTE #FIELDX = POS(FIELD1)
   VALUE ...
      COMPUTE #FIELDX = POS(FIELD2)
   ...
END-DECIDE
...
REINPUT ... MARK #FIELDX
```

If the field specified with POS is an array, a specific occurrence must be specified; for example, "POS(FIELDX(5))". POS cannot be applied to an array range.

POS and *CURS-FIELD

The system function POS*(field-name)* may be used in conjunction with the Natural system variable *CURS-FIELD to make the execution of certain functions dependent on which field the cursor is currently positioned in.

CURS-FIELD contains the internal identification of the field in which the cursor is currently positioned; it cannot be used by itself, but only in conjunction with POS(field-name)*. You may use them to check if the cursor is currently positioned in a specific field and have processing performed depending on that condition.

Example:

```
IF *CURS-FIELD = POS(FIELDX)
   MOVE *CURS-FIELD TO #FIELDY
END-IF
...
REINPUT ... MARK #FIELDY
```

Note:
The values of *CURS-FIELD and POS(field-name) serve only as internal identifications of the fields and cannot be used for arithmetic operations.

Note for Natural RPC:
If *CURS-FIELD and POS(field-name) refer to a context variable, the resulting information can only be used within the same conversation.

RET - Return Code Function

Format/length: I4

The system function RET*(program-name)* may be used to receive the return code from a non-Natural program called via a CALL statement.

RET*(program-name)* can be used in an IF statement and within the arithmetic statements ADD, COMPUTE, DIVIDE, MULTIPLY and SUBTRACT.

Example:

```
DEFINE DATA LOCAL
1 #RETURN (I4)
...
END-DEFINE
...
...
CALL 'PROG1'
IF RET('PROG1') > #RETURN
   WRITE 'ERROR OCCURRED IN PROGRAM 1'
END-IF
...
```

 Copyright © Software AG 2002

SORTKEY - Sort-Key Function

Format/length: A253

Several national languages contain characters (or combinations of characters) which are not sorted in the correct alphabetical order by a sort program or database system, because the sequence of the characters in the character set used by the computer does not always correspond to the alphabetical order of the characters.

For example, the Spanish letter "CH" would be treated by a sort program or database system as two separate letters and sorted between "CG" and "CI" - although in the Spanish alphabet it is in fact a letter in its own right and belongs between "C" and "D".

Or it may be that, contrary to your requirements, lower-case and upper-case letters are not treated equally in a sort sequence, that letters are sorted after numbers (although you may wish them to be sorted before numbers), or that special characters (for example, hyphens in double names) lead to an undesired sort sequence.

In such cases, you can use the system function SORTKEY*(character-string)* to convert "incorrectly sorted" characters (or combinations of characters) into other characters (or combinations of characters) that are "correctly sorted" alphabetically by the sort program or database system.

The values computed by SORTKEY are then only used as sort criterion, while the original values are used for the interaction with the end-user.

You can use the SORTKEY function as an arithmetic operand in a COMPUTE statement and in a logical condition.

As *character-string* you can specify an alphanumeric constant or variable, or a single occurrence of an alphanumeric array.

When you specify the SORTKEY function in a Natural program, the user exit NATUSK*nn* will be invoked - *nn* being the current language code (that is, the current value of the system variable *LANGUAGE).

You can write this user exit in any programming language that provides a standard CALL interface. The *character-string* specified with SORTKEY will be passed to the user exit. The user exit has to be programmed so that it converts any "incorrectly sorted" characters in this string into corresponding "correctly sorted" characters. The converted character string is then used in the Natural program for further processing.

For details on the user exit, see your Natural Operations documentation.

Example:

```
DEFINE DATA LOCAL
  1 CUST VIEW OF CUSTOMERFILE
    2 NAME
    2 SORTNAME
END-DEFINE
...
*LANGUAGE := 4
...
REPEAT
  INPUT NAME
  SORTNAME := SORTKEY(NAME)
  STORE CUST
  END TRANSACTION
  ...
END-REPEAT
...
READ CUST BY SORTNAME
  DISPLAY NAME
END-READ
...
```

Assume that in the above example, at repeated executions of the INPUT statement, the following values are entered: "Sanchez", "Sandino" und "Sancinto" .

At the assignment of SORTKEY(NAME) to SORTNAME, the user exit NATUSK04 would be invoked. This user exit would have to be programmed so that it first converts all lower-case letters to upper-case, and then converts the character combination "CH" to "CX" - where X would correspond to the last character in the character set used, i.e. hexadecimally H’FF’ (assuming that this last character is a non-printable character).

The "original" names (NAME) as well as the converted names to be used for the desired sorting (SORTNAME) are stored. To read the file, SORTNAME is used. The DISPLAY statement would then output the names in the correct Spanish alphabetical order:

```
Sancinto
Sanchez
Sandino
```

 Copyright © Software AG 2002

Terminal Commands

This section describes the Natural terminal commands. With terminal commands you can perform a wide variety of special-purpose functions.

- General Considerations
- Overview of Terminal Commands
- Terms Used
- CLEAR Key - Interrupt Current Operation
- CTRL+D Keys - Interrupt Current Operation
- RESET+ENTER Keys - Attention Interrupt

General Considerations

Control Character

The first character of a terminal command is the terminal command control character. It identifies the command as a terminal command. By default, the control character is "%". However, you can define another special character as control character; this is done with the session parameter CF.

Note:
When the control character is changed, all terminal commands which have been assigned to function keys will be adjusted accordingly.

Entering a Terminal Command

The following rules apply to the use of terminal commands:

- You can enter a terminal command in any unprotected field on the screen (including the message line if it is unprotected).
- A terminal command must be entered in one field, except the control character, which may be entered in a preceding field.
- It is recommended to enter a blank after a terminal command, or delete the remaining content of the field in which you enter the command. Otherwise Natural might misinterpret the field content as part of the terminal command.
- If you enter multiple terminal commands on the same screen, only the first command encountered will be executed, and all others will be ignored.
- Terminal commands which have been entered incorrectly are ignored, but you will not receive a corresponding error message.
- If you enter a terminal command and at the same time data in other fields on the screen, only the terminal command will be executed; the data will not be processed.
- Under OpenVMS and UNIX (except on Natural menu screens and selection windows), as soon as you enter the control character, a window is displayed in which you can enter a terminal command.

Using Terminal Commands in Programs

Terminal commands may also be issued from within a program by using the SET CONTROL statement. When a terminal command is specified with a SET CONTROL statement, the control character is omitted.

Overview of Terminal Commands

The following table provides an overview of the terminal commands and the functions they perform.

Command	Function
%	Continuation indicator for INPUT statements in batch programs.
%A	Re-running a recording.
%B	Activate/de-activate the recording process.
%B=	Library for recording.
%C	Copy current screen into Natural source area.
%CC, **%CS**	Copy data to the stack or into the system variable *COM.
%D	Activate keyword/delimiter mode.
%D=	Control of outlining.
%E	Display screens recorded with NATPAGE.
%E=	Activate/de-activate error processing.
%F	Activate forms/screen mode.
%F=	Frame characters for window.
%G	Set re-run mode for recording.
%H	Produce hardcopy output.
%I	Record current screen with NATPAGE.
%J	Invoke helproutine.
%K*nn*, **%KP***n*	Simulate PF- and PA-keys.
%KN, **%KO**, **%KS**	Function-key logic for Siemens terminals.
%L	Disable lower- to upper-case translation.
%L=	Set language code.
%M	Control of message line.
%MSGSF	Display system error messages in full.
%N	Activate non-conversational mode.
%O	Terminate NATPAGE.
%P	Activate NATPAGE for all subsequent screens.
%P=	CALL options.
%Q	Suppress map printing in batch mode.
%QO	Suppress pseudo-conversational output.
%QS	Simultaneous output of multiple screens.
%R	Repeat INPUT statement.
%RM	Protection of light pen sensitive fields.
%RN	Suppress compression of screen data.

 Copyright © Software AG 2002

Command	Function
%RO	Activate/de-activate screen optimization.
%S	Resume NATPAGE.
%T and **%T***ll/cc*	Set cursor position.
%T*	Position cursor outside window.
%T+ and **%T-**	Position cursor on protected field.
%TRE	Activate/deactivate external trace.
%TRI	Activate/deactivate internal trace.
%T=	Activate converter routine for specific device type.
%U	Enable lower- to upper-case translation.
%V	Control of print mode.
%W	Natural window handling.
%X	Control of statistics line/infoline.
%Y	Control of PF-key lines.
%Z	Clear source area.
%?	Invoke help information for terminal commands.
%% and **%.**	Interrupt current Natural operation.
%.P	Delete first entry from stack.
%.S	Read stack entry without deleting it.
%+ and **%-**	Enable/disable use of Natural Connection.
%*	Disable display of input characters. In batch mode, suppress printing of next input record read.
%/	Forces end-of-file condition when using INPUT statement in batch mode.
%=	Assign colors to fields.
%<TECH	Display technical information.
%<TEST	Invoke debugging facility.

Key Assignments

Key(s)	Function
CLEAR	Interrupt current Natural operation; interrupt recording.
CTRL+D	Interrupt current Natural operation.
RESET+ENTER	Terminate current processing loop.

Terms Used

In the descriptions of several terminal commands, the terms "screen" and "window" are used with the following meanings:

Screen	Depending on the operating system under which Natural is running, "screen" refers either to the entire terminal screen as such, or to the operating-system window in which the Natural session is running, or to the Natural main output window. However, for the sake of convenience, the term "screen" is used in all these instances.
Window	Always refers to the Natural window (as explained with the terminal command %W).

 Copyright © Software AG 2002

% - Continuation Indicator for INPUT in Batch

```
%
```

When a "%" is used as the last non-blank character of a data record in batch mode (applies only on mainframe computers), it causes the next record to be treated as a continuation record. See the INPUT statement in the Natural Statements documentation for further information.

%A - Re-running a Recording

```
%A name
```

This command is only available on mainframe computers.

With the terminal command **%A***name*, you cause the recording saved under that *name* to be executed again. This requires that your current library is the one in which the recording is stored.

If you issue the command **%A***name* while a session is being recorded, the recording specified with **%A***name* will not be executed but the command **%A***name* is included into the 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 **%A***name* command stops after that command and is not resumed when the execution of name finishes.

Further information on recording is provided in Recording in the section Debugging and Monitoring.

%B - Activate/De-activate the Recording Process

```
%B [name]
```

This command is only available on mainframe computers.

The recording process is activated with the terminal command **%B***name*, and de-activated with the terminal command **%B**.

With the terminal command **%B***name*, you activate the recording process. All subsequent actions will be recorded. The name you specify is the name under which the recorded data will be saved. Names of recordings must be unique within a library.

With the terminal command **%B**, you de-activate the current recording process. The recorded source is automatically saved, and you can then re-run it as often as you wish.

You may also use **%B** to insert additional actions into the recording: after you have interrupted the re-run of a recording with the CLEAR key, enter the command **%B**, and all action you perform will be inserted into the source of the recording until you enter **%B** again. Then the execution of the recording will be resumed.

Further information on recording is provided in Recording in the section Debugging and Monitoring.

%B= - Recording Library

```
%B = library-name
```

This command is only available on mainframe computers.

With the terminal command **%B=***library-name*, you specify the library in which all subsequent recording actions are to be stored.

If you activate the recording process without having specified a library, the name of the library in which the recording is stored will be 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 will remain the same (that is, either the one specified with **%B=** or the *INIT-USER library); this means that one recording may record actions across multiple applications.

Further information on recording is provided in Recording in the section Debugging and Monitoring.

%C - Copying Contents of Page Buffer

```
%C
```

This command is used to copy the contents of the page buffer to the next available lines in the Natural source work area.

The page currently displayed by Natural will be copied into the Natural source work area. The page content will be written to the next free location in the source work area, where it can be modified using the Natural editor.

To clear the source work area before copying the page, the **%Z** command may be used.

Notes:
%C should not be used in an editor session. Modifications made to the source area outside of editor content are not recognized by the editor.

The page buffer (the logical output from Natural) is not necessarily the same as the screen buffer which is displayed on the screen.

Example:

```
FOR I = 1 TO 10    WRITE I    SET CONTROL 'C' END-FOR END
```

 Copyright © Software AG 2002

%CS and %CC - Copying Data to Stack or *COM

```
%C {C} {C   } [A    ] [S]
   {S} {Lnn} [I    ]
             [color]
             [1    ]
             [W    ]
```

* **1** and **W** cannot be specified with **L***nn*.

With this terminal command, you can copy parts of a screen into the Natural stack (%CS) or into the system variable *COM (%CC). The protected data from a specific screen line are copied field by field (except with option "A"; see below).

The second letter in the command determines where the data are copied to:

- **%CC...** copies the data into the system variable *COM.
- **%CS...** copies the data into the Natural stack. The data are placed on top of the stack as input data (as with a STACK TOP DATA statement).

The third letter in the command determines the line from which the data are copied:

- **%CCC** and **%CSC** copy all protected data from the line in which the cursor is positioned, beginning from the field in which the cursor is positioned.
- **%CCL***nn* and **%CSL***nn* copy all protected data from line number nn.

Moreover, you have the following options:

- **%C...A** copies all of a line, that is, not only the protected data, but also the modifiable fields; the line is not copied field by field, but as a whole (including field attributes).
- **%C...I** copies only the intensified fields from a line.
- **%C...***color* copies only the fields of that color from a line.
- **%C...C1** copies only one field, namely the field in which the cursor is positioned (regardless of its attributes). (**%C...L***nn***1** is not possible.)
- **%C...CW** copies only the word (as delimited by blanks or special characters within a field) over which the cursor is positioned. (**%C...L***nn***W** is not possible.)
- **%C...S** causes Natural to "stay" on the screen from which the data are copied, when the command is executed. This allows you to copy several different data from a screen, before you process the data.

When you enter the command directly as %C... (or assign it to a PF-key), it applies to the *physical screen* within the active window. Exception: On mainframe computers, it applies to the entire *physical screen*; this means that you can copy from the entire screen, no matter whether the data you wish to copy are inside or outside an active window.

When you issue the command via a SET CONTROL statement, it applies to the *logical page* built by Natural. In conjunction with the stay option, this allows you to copy all data you need from an entire logical page (which may be larger than the actual physical screen), before you process these data.

%D - Activate Keyword/Delimiter Mode

```
%D
```

This command is only available on mainframe computers.

This command is used to activate keyword/delimiter mode. This mode is recommended for processing in batch mode. See the INPUT statement in the Natural Statements documentation for additional information.

%D= - Control of Outlining

```
%D = B { M
         I
         color-code
         B
         R
         W } ...
```

This command is only available on mainframe computers.

The outlining feature is only available on certain types of terminals, usually those which also support the display of double-byte character sets.

The terminal command %D=B is used to control outlining.

Outlining (boxing) is the capability to generate a line around certain fields when they are displayed on the terminal screen. Drawing such "boxes" around fields is another method of showing the user the lengths of fields and their positions on the screen.

The %D=B command provides the following options:

 Copyright © Software AG 2002

%D=BM	This command causes all physically modifiable fields (AD=M or AD=A) to be displayed with outlining. If this command is repeated, this feature will be disabled again: in other words, it will return to the default state.
%D=BI	This command causes all intensified fields (AD=I) to be displayed with outlining.
%D=B*color-code*	This command causes all fields of the specified color to be displayed with outlining. For valid color codes, see the session parameter CD.
%D=BB	This command causes "big boxes" to be generated: if several fields displayed underneath one another have identical display characteristics, one big box is drawn around all of them. Normally (if %D=BB is not specified) an individual box is drawn around each individual field.
%D=BR	This command causes all boxes to be reset; that is, no field will be displayed with outlining.
%D=BW	This command causes all windows which are to be displayed with frame (see the DEFINE WINDOW statement in the Natural Statements documentation) to be displayed with outlining instead of frame.

When outlining is active, filler characters in modifiable fields are not displayed, because they serve the same purpose as outlining, namely showing the user the position and length of a field.

%E - Display Screens Recorded with NATPAGE

```
%E
```

This command is not available under Windows 98 and Windows NT/2000.

With this command, you can display any screens that have been recorded with the NATPAGE utility.

This command suspends normal processing and enters the display mode of NATPAGE: A list of all screens recorded with NATPAGE (by using the terminal commands %I and %P) is displayed; including screen number, time of recording, and map name (if the screen is a map). From the list you can then select the screens you wish to have displayed. You cannot enter any data on the displayed screens.

When a screen is recorded with %I or %P, two items of information are superimposed on the screen: the time when the screen was recorded, and a screen number (the screens are numbered sequentially in the order in which they are recorded).

On mainframe computers, in NATPAGE display mode, an input field (CMD) is provided in which you can enter the following scrolling commands:

TOP or **T**	Displays the first screen which was recorded.
BOT or **B**	Displays the last screen which was recorded.
nnn	Displays the screen with screen number nnn.
+*nnn*	Scrolls nnn screens forward.
-*nnn*	Scrolls nnn screens backward.
.	Terminates display mode.

If you attempt to scroll to a screen number which does not exist, either the first or the last screen is displayed, depending on the scrolling direction.

Under OpenVMS and UNIX, a window will be displayed in which you can scroll through the list of recorded screens. With ENTER, you can select a screen for display. With ESC, you leave the function.

See also the terminal commands %P, %O, %S, and %I.

%E= - Activate/De-activate Error Processing

```
%E = { ON  }
     { OFF }
```

With the terminal command %E=OFF, you switch any error transaction (as identified by the system variable *ERROR-TA or defined in Natural Security) and ON ERROR processing off; any error that occurs will then be handled by normal Natural error processing.

This may be used to locate an error in your applications' error processing if the structure of the application with various error-handling procedures on different levels makes it impossible for you to find out where exactly an error originally occurred.

With the terminal command %E=ON, you switch error transaction and ON ERROR processing on again.

%F - Activate Forms/Screen Mode

```
%F
```

This command is only available on mainframe computers.

This command is used to activate forms/screen mode. See the INPUT statement in the Natural Statements documentation for additional information on forms/screen mode.

%F= - Frame Characters for Window

```
%F = chv
```

With this terminal command, you can define the characters to be used in the frame of a window.

Note: On graphical user interfaces, this command is ignored.

c	The first character will be used for the four *corners* of the window frame.
h	The second character will be used for the *horizontal* frame lines (top and bottom frame lines).
v	The third character will be used for the *vertical* frame lines (left and right frame lines).

 Copyright © Software AG 2002

For example, %F=+-! would cause the window frame to look like this:

```
+-----------------------+ !                    ! !                    ! !                    ! !                    ! +-----------------------+
```

For further information on window processing, see the DEFINE WINDOW statement in the Natural Statements documentation.

%G - Re-run Mode for Recording

```
%G [ON]
   [OFF]
```

This command is only available on mainframe computers.

A recording can be re-run in two modes: background mode and film mode.

In background mode, the entire recording is re-run invisibly; that is, all the actions of the recording are carried out without anything being displayed to you on the terminal screen during the execution of the recording. Neither can you interrupt a recording that is re-run in background mode (unless the recording contains the terminal command %R).

In film mode, a recording is re-run step by step, and all actions are displayed to you. By pressing ENTER you proceed from one step to the next. In film mode, it is also possible for you to interrupt the recording by pressing the CLEAR key.

By default, a recording is re-run in background mode.

%GON	With this command, you switch film mode on.
%GOFF	With this command, you switch back to background mode.
%G	With this command, you toggle between the two modes.

Further information on recording is provided in Recording in the section Debugging and Monitoring.

%H - Hardcopy Output

```
%H [!] [ destination ]
       [ ,          ]
       [ .          ]
       [ #destination ]
       [ = [destination] ]
       [ —              ]
```

This command is used to produce hardcopy output (valid only if the hardcopy feature is implemented).

The **%H** command produces output from Natural reports and communication screen layouts on a hardcopy printer.

By default, a %H command is effective for the current logical output (that is, the current window without message line, function-key lines and statistics line/infoline). However, you can also print the current screen content (see **%H,** and **%H.** below).

You have the following options:

%H	A window is displayed listing all available printers; from this list you select the printer on which the hardcopy is to be printed. On mainframe computers, no selection window is displayed; instead, the printer specified with the profile parameter HCDEST (Natural Operations for Mainframes documentation) is used. Under Windows, printer "LPT1" will be used by default if you do not specify one.
%H*destination*	The hardcopy will be output on the specified *destination*. The *destination* can be 1 to 8 characters long.
%H! and **%H!***destination*	These commands are only valid under UNIX. SET CONTROL 'H!*destination*' is valid only for the current page (that is, the page that was output before the SET CONTROL statement was executed). %H! applies to the page currently displayed on the screen.
%H,	All subsequent %H commands apply to the current *screen* (only possible on mainframe computers).
%H.	All subsequent %H commands apply to the current *logical page* (applies by default).
%H=	Every time ENTER is subsequently pressed, a hardcopy is produced (toggle switch).
%H=*destination*	With **%H=***destination*, you specify the hardcopy device to which *all* logical output is to be output. This feature may be used, for example, to log a sequence of output for administrative, debugging or educational purposes.
%H#*destination*	With **%H#***destination*, you can have the hardcopy output routed to a special destination, as defined with a DEFINE PRINTER statement, for example, to the Natural source area, to Con-nect, or to the INFOLINE.
%H-	Stops hardcopy printing immediately. **Note:** When a SET CONTROL 'H-' statement is executed, data which were already written to the page buffer but have not been output yet, will not be routed to the printer. To also print these data, you have to code an EJECT statement before the SET CONTROL 'H-' statement.

Generally, a %H command (entered anywhere on the screen or used with a SET CONTROL statement) is valid until either the next INPUT statement with modifiable fields is executed, or until the end of the program. However, when %H is entered in an input field, only the current logical page will be printed.

Natural under CMS: Specifying %HL produces a file called NATURAL LISTING A.

 Copyright © Software AG 2002

%I - Record Current Screen with NATPAGE

```
%I
```

This command is not available under Windows 98 and Windows NT/2000.

When you enter this command on a screen, the screen will be recorded by the NATPAGE utility.

The maximum number of screens that can be recorded with NATPAGE is determined by the session parameter PD. If this number is exceeded, each additional screen will overwrite an already recorded one, beginning with the one recorded first.

To display screens that have been recorded with NATPAGE, you use the terminal command %E.

See also the terminal commands %P, %O, %S, and %E.

%J - Invoke Helproutine

```
%J helproutine
```

This terminal command can be used to invoke an interactive helproutine.

When %J is used after interrupting the re-run of a recording (applies only on mainframe computers), the re-run will be resumed after the helproutine has been executed (refer to Recording in the section Debugging and Monitoring for details on recordings).

Note:
This terminal command should not be issued when a function invoked by a system command is active.

%K and %KP - Simulate PF- and PA-Key

```
%K {nn}
   {Pn}
```

%K*nn* simulates the terminal function key numbered *nn* (PF1 to PF24). This permits PF-keys 13-24 to be assigned to PF-keys 1-12, or the activation of PF-keys not available on the keyboard used. On mainframe computers, it also makes function keys available in batch mode.

%K0 simulates the ENTER key.

%KP*n* simulates the program attention key numbered *n* (PA1 to PA3) (see %K*nn*).

%KN/%KO/%KS - Siemens Function-Key Logic

```
%K {N} [N]
   {O}
   {S}
```

These commands are only applicable to Siemens terminals.

%KN	For terminal types 8160, 974n, 9750 - 9755, this command causes the literals "%K1" to "%K20" to be loaded to the keys P1 to P20. For terminal types 9756, 9758, 976n, this command causes the send-key codes "F1" to "F20" to be loaded to the keys P1 to P20.
%KO	This command causes the literals "01" to "20" as well as the send-key code "F5" to be loaded to the keys P1 to P20.
%KS	This command causes the literals "A" to "T" as well as the send-key code "F5" to be loaded to the keys P1 to P20.

If an "N" is specified after the respective terminal command (that is, %KNN, %KON, or KSN), only the corresponding function-key mode (KN, KO or KS mode) will be activated, but no values will be loaded to the P-keys.

See also Natural under BS2000/OSD in the Natural Operations for Mainframes documentation.

%L - Disable Lower- to Upper-Case Translation

```
%L
```

This command prevents that lower-case characters are translated to upper-case by Natural.

On a mainframe computer, you should also ensure that the TP monitor used does not perform any translation before data are passed to Natural.

See also the terminal command %U.

%L= - Language Code

```
%L = nn
```

With the terminal command **%L=***nn*, you can set the language code nn to be used by Natural. For a list of possible language codes, see the system variable *LANGUAGE.

 Copyright © Software AG 2002

%M - Control of Message Line

```
%M [ T
     B
     [-]nn
     P
     ={BL
       GR
       NE
       PI
       RE
       TU
       YL} ]
```

This command is ignored in batch mode.

With this terminal command, you can control the Natural message line.

Message Line Positioning:	
%MB	causes the message line to be the bottom line of the screen.
%MT	causes the message line to be the top line of the screen.
%M	causes the current message line position to be switched from the top line to the bottom line of the screen (or vice versa), or from line *nn* to the bottom line.
%M*nn*	causes the message line to be placed on line *nn* of the screen.
%M-*nn*	causes the message line to be placed on the *nn*th line from the bottom of the screen. If the line number *nn* or -*nn* is not within the current screen, the message line will not be displayed.
Message Line Protection:	
%MP	switches from protected to unprotected message line or vice versa.
Message Line Color:	
%M=*color-code*	causes the message line to be displayed in the specified color (for explanation of color codes, see the session parameter CD).

Note:
This terminal command also applies to Natural's NEXT (or MORE) line.

%MSGSF - Display System Error Messages in Full

```
%MSGSF = { ON  }
         { OFF }
```

This command is only available on mainframe computers.

By default, a Natural system error message consists of: the name of the program and the number of the line that caused the error, followed by the actual text of the message. Depending on the size of the window in which the message is displayed, the actual text may be truncated. With the terminal command %MSGSF, you can avoid such truncation.

%MSGSF=ON	System error messages will be displayed in full; that is, program name, line number and actual message text will be displayed (this is the default).
%MSGSF=OFF	System error messages will be displayed in short form; that is, only the actual message text will be displayed (but not the program name and line number).

Instead of "ON", you can also specify "+" or "T" (true);
instead of "OFF", you can also specify "-" or "F" (false).

Note:
The display format of system error messages can also be controlled with the profile parameter MSGSF (see the section Profile Parameters in the Parameter Reference documentation).

%N - Activate Non-Conversational Mode

```
%N
```

This terminal command is used with a SET CONTROL statement and causes the next screen to be displayed without requiring any user response for processing to continue; that is, after the screen has been displayed, processing will continue immediately without waiting for any user input.

This command may be used to send messages about the progress of program execution to the user.

The command only applies to the next output screen (that is, the screen that is output when the next I/O is executed).

Note:
Under IMS/TM, this terminal command does not apply if an IMS scratch-pad area is used.

%O - Terminate NATPAGE

```
%O
```

 Copyright © Software AG 2002

This command is not available under Windows 98 and Windows NT/2000.

With this command, you terminate the recording of screens by the NATPAGE utility (as activated by the terminal command %P). The current screen is recorded. All screens recorded since the last %P command are retained.

See also the terminal commands %E, %I, %P and %S.

%P - Activate NATPAGE for all Subsequent Screens

```
%P
```

This command is not available under Windows 98 and Windows NT/2000.

With this command, you activate the utility NATPAGE to record the current screen and all subsequent screens.

The maximum number of screens that can be recorded with NATPAGE is determined by the session parameter PD. If this number is exceeded, each additional screen will overwrite an already recorded one, beginning with the one recorded first.

All screens that have been recorded with previous %P and %I commands will be deleted when you issue a %P command.

To display screens that have been recorded with NATPAGE, you use the terminal command %E.

See also the terminal commands %E, %I, %O and %S.

%P= - CALL Options

```
%P = { S }
     { V }
     { C }
     { L }
```

These commands only apply on mainframe computers.

The commands %P=S, %P=V and %P=C may be used to set special options which apply when a Natural program calls a non-Natural program (via a CALL statement) under CICS. In all other environments, these commands are ignored.

For details on calling non-Natural programs, see the CALL statement in the Natural Statements documentation.

Each %P= command applies only to the next call.

%P=S - Standard Linkage for Call

This command only applies under CICS.

Normally, when a Natural program calls a non-Natural program under CICS, the call is accomplished by an "EXEC CICS LINK" request.

If standard linkage is to be used for the call instead, issue the terminal command %P=S. In this case, the called program must adhere to standard linkage conventions with standard register usage.

%P=V - Roll-Out for Call

This command only applies under CICS.

Normally, when a Natural programs calls a non-Natural program and the called program issues a conversational terminal I/O, the Natural thread is blocked until the user has entered data.

Under CICS, %P=V can be used to prevent the Natural thread from being blocked: if this terminal command is specified, the parameter data passed from the Natural program to the called program are copied out of the Natural thread, and the thread is rolled out before the call. The thread is then available for another user. Upon return from the called program to the calling Natural program, the thread is rolled back in again, the (modified) data area is copied into the thread, and Natural processing continues.

Note:
Only the parameters specified in the CALL statement are copied out of and back into the thread.

%P=C - Pass Parameter Values instead of Address

This command only applies under CICS.

Normally, when a Natural program calls a non-Natural program under CICS, the address of the CALL parameter address list is passed in the COMMAREA. If you wish the parameter values themselves, rather than the address of their address list, to be passed in the COMMAREA, issue the terminal command %P=C before the call.

This makes it possible, for example, to use DPL for called CICS programs: A CICS program that resides in another CICS region can only be called with %P=C; as addresses within the "calling" region cannot be accessed by the "called" region, the parameter values have to be passed instead.

When %P=C is used, no parameters are passed in the TWA, but only parameter values in the CICS COMMAREA. All parameters of the CALL parameter list are copied adjacent to one another, regardless of their alignment. The resulting COMMAREA length is the sum of the individual parameter lengths (this has to be taken into consideration when determining the number of array occurrences to be passed). On return from the called program, the parameters are copied back.

When overlapping fields are passed or the same field is passed more than once, these fields should be made "read-only" for the called program; otherwise, unpredictable results may occur when parameter values are returned to the calling program.

For %P=C, the restriction applies that group arrays cannot be passed:

```
01 #GROUP (2)    02 #FIELD1 (A1)    02 #FIELD2 (P7)
```

Either pass them as individual arrays:

```
01 #GROUP    02 #FIELD1 (A1/2)    02 #FIELD2 (P7/2)
```

 Copyright © Software AG 2002

Or redefine them:

```
01 #GROUP 01 REDEFINE #GROUP    02 #ARRAY (A1/10)
```

and specify the array name in the CALL statement.

Note:
If both %P=S and %P=C are issued for a call, %P=C will be ignored.

%P=L - Call of LE/370 Dynamic Main Program

This command only applies if the option for the support of the IBM Language Environment (LE/370) calling conventions has been set at the installation of Natural.

This command causes control to be returned to Natural after a call to an LE/370 dynamic main program.

By default, when an LE/370 dynamic main program is invoked, control is not returned to Natural after the invoked program has been processed. For control to be returned after that program has been processed, you have to use %P=L before invoking the program.

For information on the support of LE/370 calling conventions, see the section Operating Natural - Environment-Specific Information in the Natural Operations for Mainframes documentation.

%Q - Suppress Map Printing in Batch Mode

```
%Q
```

In interactive processing, the %Q command is ignored.

When issued in batch mode (applies only on mainframe computers), the %Q command causes the output of maps, or screens produced by INPUT statements, to be suppressed.

SET CONTROL 'Q' (online or batch) causes the next INPUT statement *not* to be processed. This may be used, for example, if at the end of a helproutine the processing is to continue without the user having to press ENTER upon return from the help to the map.

%QO - Suppress Pseudo-Conversational Output

```
%QO
```

This command is only applicable under CICS.

In a Natural session running under CICS in pseudo-conversational mode, you can switch to another CICS transaction by issuing the statement CALL 'CMTASK'. To enable CICS to start the other transaction, a terminal I/O, i.e. an INPUT statement, is required after the CALL statement.
To suppress the screen output of that INPUT statement (which would be overwritten immediately by the started transaction anyway), you issue a SET CONTROL 'QO' statement before the INPUT statement.

%QS - Simultaneous Output of Multiple Screens

```
%QS
```

With this command, you can display multiple screens simultaneously.

%QS causes the next screen I/O not to be executed. The corresponding output screen is kept internally until the following I/O, when it is displayed together with the next screen.

Therefore %QS only makes sense if the second output screen is a window, that is, if it does not entirely overlay the first screen that was suppressed with %QS.

For example, you can suppress the output of a screen A with %QS; the next screen B is a window which partially overlays screen A (perhaps a help window for one of the fields on screen A); with the next screen I/O, the window B and the "underlying" screen A are displayed simultaneously.

A %QS command only applies to the subsequent screen.

Note:
As %QS reduces the number of screen I/Os, it also improves performance.

%R - Repeat INPUT Statement

```
%R
```

This command causes INPUT statement repetition and the output screen to be rebuilt. All output data generated from the beginning of the INPUT statement will be reproduced.

%R in a Recording - on Mainframe Computers only

By recording the terminal command %R, you can manipulate a single step in a recording when it is re-run.

You may also use %R to overwrite input data in a recording that is being re-run.

Further information on recording is provided in Recording in the section Debugging and Monitoring.

 Copyright © Software AG 2002

%RM - Protection of Light Pen Sensitive Fields

```
%RM
```

This command causes all light pen sensitive fields on the screen to be made write-protected, that is, the user can select them with a light pen, but not overwrite their contents. To switch the write-protection off again, issue the command again.

For a field to be light pen sensitive, it must be displayed intensified (AD=I) or blinking (AD=B), and the first character in the field must be one of the light pen designator characters ("?", ">", "&" or a blank); selecting a field with a light pen causes the designator character to be changed, therefore you can make the processing of fields selected with a light pen dependent on the values of the designator characters.

%RN - Suppress Compression of Screen Data

```
%RN
```

This command is only available on mainframe computers and under OpenVMS and UNIX.

For the next screen I/O, this command suppresses Natural's automatic compression of screen data and causes the entire screen to be sent instead.

At a screen I/O, Natural usually does not send the entire screen, but only the changed screen data. When a non-Natural program is invoked which causes a screen I/O, Natural in most TP environments recognizes the non-Natural I/O and sends the entire screen upon the next screen I/O. However, in some TP environments, the non-Natural I/O is not recognized by Natural; in such a case, %RN should be used.

%RN only applies to the next screen I/O.

%RO - Activate/De-activate Screen Optimization

```
%RO [ON]
    [OFF]
```

This command is only available on mainframe computers.

With this command you can switch Natural's automatic screen optimization off and back on again.

Normally, Natural's screen optimization causes screens to be sent as compressed as possible. If this should conflict with any TP monitor's screen optimization or hardware limitations, you can use this terminal command to switch Natural's screen optimization off; screens are then sent in non-compressed form.

If you use the session parameter BX setting BX=L or BX=R, you should switch off Natural's screen optimization.

%ROON	activates screen optimization.
%ROOFF	deactivates screen optimization.
%RO	switches between the two modes.

Instead of "ON", you can also specify "+" or "T" (true).
Instead of "OFF", you can also specify "-" or "F" (false).

%S - Resume NATPAGE

```
%S
```

This command is not available under Windows 98 and Windows NT/2000.

This command is used to resume the recording of screens by the NATPAGE utility.

The recording of screens by NATPAGE (which is activated with the terminal command %P) can be interrupted with the terminal command %O. After it has been interrupted, you can use the terminal command %S to resume the recording of screens by NATPAGE.

See also the terminal commands %E, %I, %O and %P.

 Copyright © Software AG 2002

%T - Position Cursor to Top of Screen

```
%T
```

This command positions the cursor at the top lefthand corner of the screen on the next screen output.

Note:
On graphical user interfaces, this command only works if there is an input field (AD=A or AD=M) at the top lefthand corner.

%T*ll/cc* - Position Cursor to Line ll, Column cc

```
%T///cc
```

This command positions the cursor at line *ll*, column *cc* on the next screen output.

The line and column positions are counted beginning with 1 within the current physical screen window. The message line, the function-key lines, and the statistics line/infoline (if active) are not counted as lines in the current window; that is, "%T1/1" always positions to the beginning of the topmost data line in the window.

Notes:
On some devices, you have to execute the terminal command %T+ before you can place the cursor at any desired position on the screen.

On graphical user interfaces, %Tll/cc only works if there is an input field (AD=A or AD=M) at the specified position.

%T* - Position Cursor Outside Window

```
%T *
```

Normally, when a window is active and the window contains no input fields (AD=A or AD=M), the cursor is placed in the top left corner of the window.

This terminal command causes the cursor to be placed in a *COM system variable outside the window when the active window contains no input fields.

%T*	Switches between cursor placement in system variable *COM outside the window and standard cursor placement within the window.

%T* only applies to the next INPUT statement, and it must be issued *before* the INPUT statement.

%T+/%T- - Position Cursor on Protected Fields

```
%T {+}
   {-}
```

These commands are only applicable to OpenVMS, UNIX and Siemens terminals.

%T+	After you issue this command, you can position the cursor anywhere on the screen, including protected fields (however, the protected fields remain write-protected). This may be useful, for example, for action bar processing (see the terminal command %YC).
%T-	With this command, you cancel the effect of the command **%T+**.

 Copyright © Software AG 2002

%TRE - Activate/Deactivate External Trace

```
%TRE {ON}
     {OFF}
```

This command is only available on mainframe computers. Do not use this command without prior consultation of Software AG support.

This command is used to activate/deactivate the external trace function. This function is intented primarily for Software AG internal use for debugging purposes. It writes trace data to an external trace dataset depending on the TP environment in which Natural is running.

Instead of "ON", you can also specify "+" or "T" (true).
Instead of "OFF", you can also specify "-" or "F" (false).

Note:
The external trace function can also be activated/deactivated with the profile parameter ETRACE (see Natural Operations documentation for Mainframes).

%TRI - Activate/Deactivate Internal Trace

```
%TRI {ON}
     {OFF}
```

This command is only available on mainframe computers. Do not use this command without prior consultation of Software AG support.

This command is used to activate/deactivate the internal trace function. This function is intended primarily for Software AG internal use for debugging purposes. It passes trace data to the SYSRDC utility which is described in the section Debugging and Monitoring.

Instead of "ON", you can also specify "+" or "T" (true).
Instead of "OFF", you can also specify "-" or "F" (false).

Note:
The internal trace function can also be activated/deactivated with the profile parameter ITRACE (see the section Profile Parameters in the Parameter Reference documentation).

%T= - Activate Converter Routine for Device Type

```
%T = { 3270 }
     { 3278 }
     { 3279 }
     { 9750 }
     { 9755 }
     { 9756 }
     { 9763 }
     { TTY  }
```

This command is only relevant if Natural is used on mainframe computers under certain TP monitors.

This command causes Natural to use the appropriate converter routine for the specified device type.

If the TP environment is not able to inform Natural of the physical terminal type it is working with (in which case the terminal type would be set automatically), this command may be used to specify the terminal type. This allows Natural to build the appropriate attribute sequence to operate the terminal.

The default terminal types used by Natural (if no information is passed from the TP environment) are %T=3270 (for IBM), or the value defined in PDN unless overridden by the parameter T975X (for Siemens; see Installing the Natural TIAM Interface in the Natural for Mainframes Installation Guide).

Instead of the %T= command, you can also use the profile parameter Terminal Type TTYPE (see the section Profile Parameters in the Parameter Reference documentation).

Note:
The above diagram lists only the most commonly used terminal types. If your terminal is of another type, ask your Natural administrator to check the NATCONFG module to ascertain whether the terminal type can be specified with %T=.

%U - Translate Lower to Upper Case

```
%U
```

This command causes Natural to translate lower-case characters to upper-case for alphanumeric input data.

Upper-case translation is in effect by default. See also the terminal command %L.

 Copyright © Software AG 2002

%V - Control of Print Mode

```
%V [ON]
   [OFF]
   [N]
   [A]
   [K]
```

This terminal command only applies to terminals which support inverted (right-to-left) print mode. The options "N", "A" and "K" are only available on mainframe terminals.

%VON	activates inverted (right-to-left) print mode.
%VOFF	activates normal (left-to-right) print mode.
%V	switches from normal to inverted print mode, and vice versa.
%VN	switches between the normal print mode and inverted print mode for numeric fields. This may be used for sensitive terminals which do not invert numerics.
%VA	causes alphanumeric fields that are defined with PM=I to be treated as numeric fields. This ensures that numeric values in such fields, which would otherwise be interpreted from right-to-left, will be interpreted correctly from left-to-right. Entering %VA again switches off the effect of %VA again.
%VK	Numeric fields for which a helproutine or edit mask is defined, are internally converted by Natural to alphanumeric format (otherwise a user would not be able, for example, to enter a question mark in such a field to invoke the helproutine). With inverted print mode, however, this internal conversion would cause a numeric value in such a field to be interpreted from right-to-left. For numeric fields that are defined with PM=I and have a helproutine or edit mask, %VK prevents the internal translation to alphanumeric format, thus ensuring the correct interpretation of values in these fields, while an edit mask will still be applied correctly and a question mark can nonetheless be entered in the field. Entering %VK again switches off the effect of %VK again.

For additional information on print modes, see the session parameter PM.

%W - Window Processing

```
%W
```

It is strongly recommended that the DEFINE WINDOW statement be used instead of the %W command.

A Natural window is that segment of a logical page, built by a Natural program, which is displayed on the terminal screen.

The %W command controls the processing of this window.

The command must always be specified with parameters for the various functions as described hereafter. Multiple parameters may be specified with one %W command; they must be specified consecutively without any delimiter characters.

There is always a window present, although you may not be aware of its existence: unless specified differently (with a %W command or DEFINE WINDOW statement), the size of the window is identical to the physical size of your terminal screen. See also the DEFINE WINDOW statement in the Natural Statements documentation for information on window processing.

There are two types of window commands:

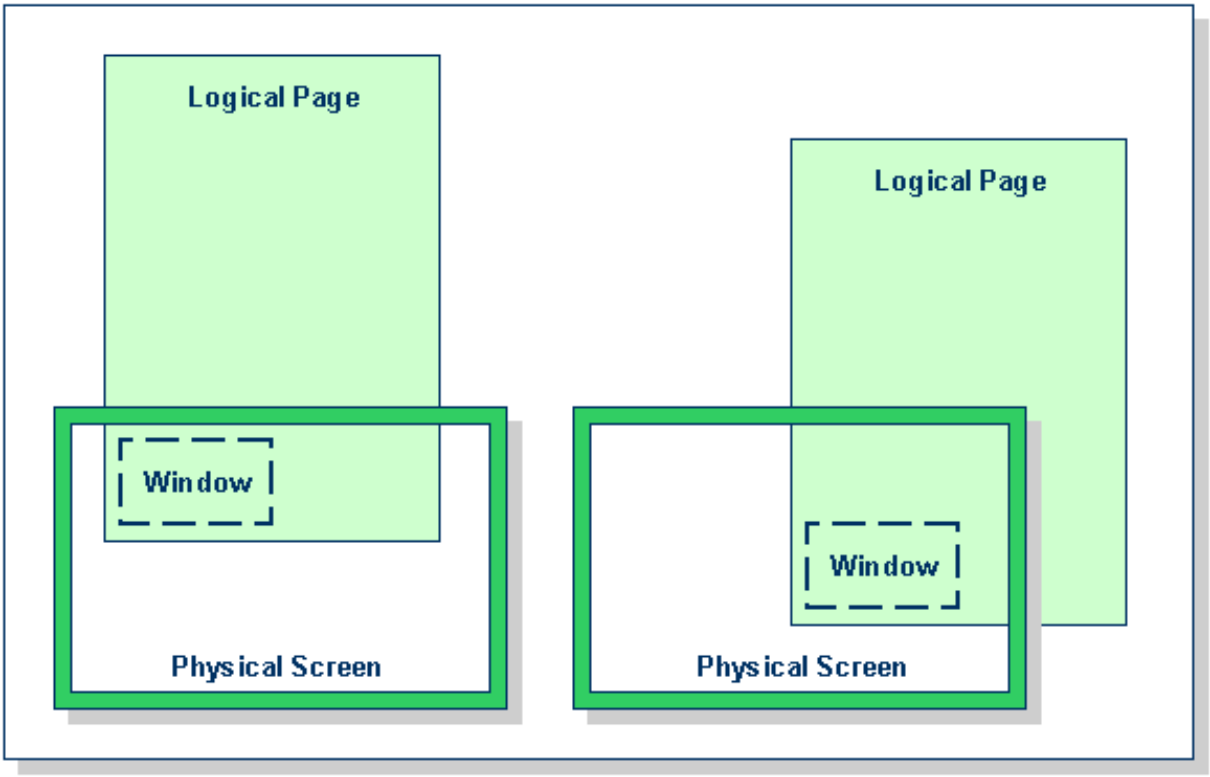
1. commands to control the size and position of the window on the physical screen;
2. commands to control the position of the window on the logical page created by the program.

 Copyright © Software AG 2002

1. Window Size and Position on the Physical Screen

The following window commands control the size and position of the window on the physical screen.

When you change the position of the window on the physical screen, the position of the window on the logical page will remain unchanged:



For information on possible window sizes, see the DEFINE WINDOW statement.

Command	Function
%WB	The window size (excluding frame) will be set to physical screen size. If a frame is defined, it will not be visible.
%WB*lll/ccc*	The top left corner of the window will be positioned to line number *lll*/column number *ccc* (lines and columns are counted on the physical screen). The window size will remain unchanged. If the window is too large to be placed at the specified position, it will be placed as close as possible to that position.
%WB0	The window will be positioned to the top left corner of the screen. The window size will remain unchanged.
%W#	The top left corner of the window will be positioned to the cursor position. The window size will remain unchanged. If the window is too large to be placed at the specified position, it will be placed as close as possible to that position.
%W?	The bottom right corner of the window will be set to the cursor position. The top left corner of the window will remain unchanged, and the size of the window will be adjusted accordingly.
%WL*nn*	The line size (horizontal extension) of the window (including frame, if specified) will be set to *nn*. If *nn* is omitted or specified larger than would fit on the screen, the line size will be set to the maximum possible (that is, to the right edge of the screen).
%WC*nn*	The column size (vertical extension) of the window (including frame, if specified) will be set to *nn*. If *nn* is omitted or specified larger than would fit on the screen, the column size will be set to the maximum possible (that is, to the bottom of the screen).

Column size and line size specifications refer to the overall physical size of the window (including frame, if specified), not to the size of what is logically visible inside the window.

Any incorrect/impossible size or positioning commands will either be ignored or adjusted to the physical possibilities.

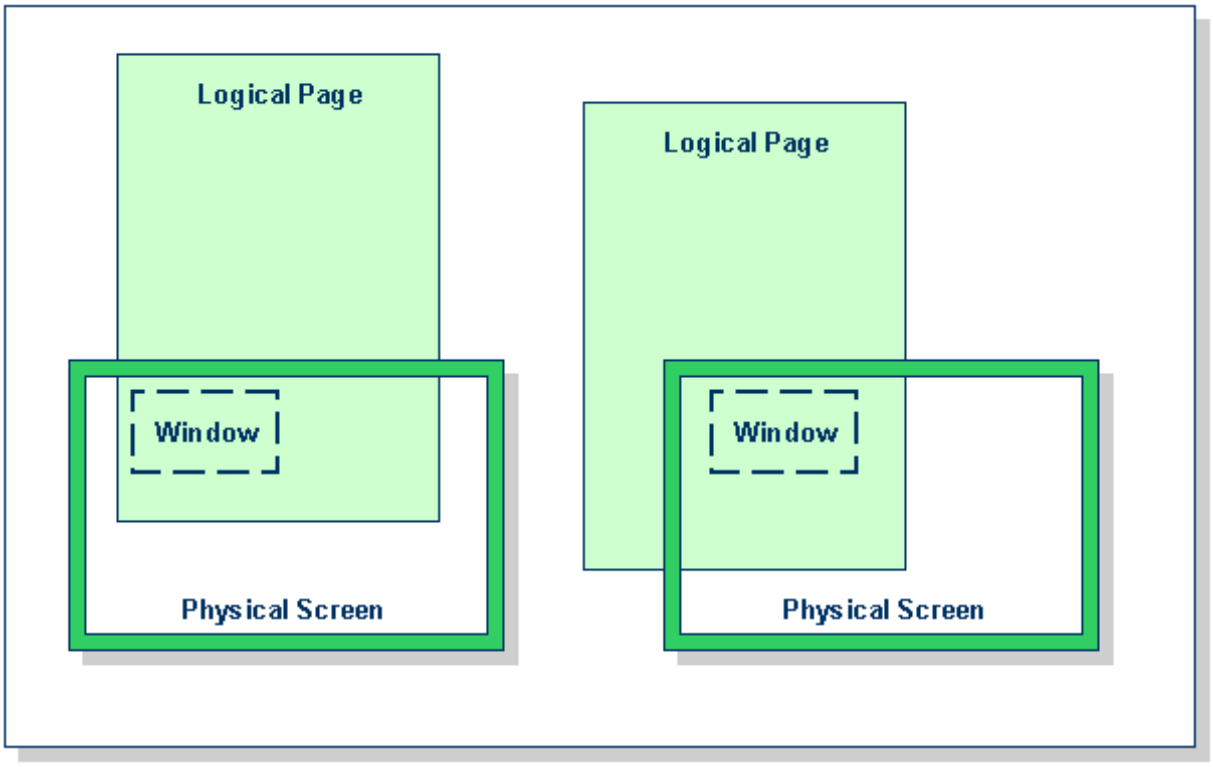
Command	Function
%WF	Switch on framing. The boundaries of the window will be indicated by a frame. If the window size is smaller than 4 lines by 12 (or 13 on mainframe computers) columns, the frame will not be visible.
%WM	Switch off framing. The boundaries of the window will not be indicated by a frame. Switching off the frame does not change the size of the window (only the size of the page segment visible inside the window).
%WO	Suppress display of PF-key lines, message line and statistics line. This command only applies if the screen is a "real" window (that is, smaller than the physical screen). To cancel the effect of %WO, you issue %WO again (or %WD).
%WP	By default, the PF-key lines, the message line and the statistics line are displayed within a window. To display them on the screen outside the window, use %WP. To cancel the effect of %WP, you use %WD.
%WD	Cancels the effects of %WF, %WO and %WP (and of the TITLE option of the DEFINE WINDOW statement).
%WX	If there is a *COM field outside the window, this field is normally not write-protected. With %WX you can make it write-protected.
%WY	Cancels the effect of %WX.

 Copyright © Software AG 2002

2. Window Position on a Logical Page

The following window commands control the positioning of the window on the current logical page, that is, the current report/map produced by the Natural program for display. This logical page may be larger in size than the physical screen.

When you change the position of the window on the logical page, the size and position of the window on the physical screen will remain unchanged. In other words, the window is not moved over the page, but the page is moved "underneath" the window:



Unless specified differently by one of the following commands, the window will be placed at the top left corner of the logical page.

Command	Function
%W*	The position marked on the page by the cursor will be shifted to the top left corner of the window.
%W*lll,ccc*	The position of the logical page determined by line number *lll*/column number *ccc* will be shifted to the top left corner of the window. Lines and columns are counted on the logical page.
%W<	Shift window left. The number of positions shifted is equal to the line size (horizontal extension) of the window.
%W<<	Shift window to leftmost position of page.
%W<*n*	Shift window left *n* positions (0 ≤ *n* ≤ logical line size).
%W>	Shift window right. The number of positions shifted is equal to the line size (horizontal extension) of the window.
%W>>	Shift window to rightmost position of page.
%W>*n*	Shift window right *n* positions (0 ≤ *n* ≤ logical line size).
%W+	Shift window down. The number of lines shifted is equal to the number of lines in the window. (*)
%W++	Shift window to bottom of page. (*)
%W+*n*	Shift window down *n* lines (0 ≤ *n* ≤ logical page size). (*)
%W-	Shift window up. The number of lines shifted is equal to the number of lines in the window.
%W--	Shift window to top of page.
%W-*n*	Shift window up *n* lines (0 ≤ *n* ≤ logical page size).
%WH	By default, the position of the window on the logical page is reset to "top left corner" after a screen I/O. %WH prevents the window position from being reset by the next I/O, that is, the set window position will be retained. %WH only applies to the next I/O.
%WS	Switch on the STAY option; that is, control will "stay" on the current page until the end of the page. If a page is not yet completely shown in vertical direction, the indicator "VVVV" will appear in the message line. The window will be scrolled downward with every ENTER until the end of the logical page is reached. The next ENTER will cause control to be returned to the program. (This does not apply to pages created by an INPUT statement with input fields (AD=A or AD=M).)
%WN	Switch off the STAY option. When you press ENTER, control is returned to the program.

* The window can be shifted to the last non-blank line of the page at most.

Note:
If you wish to use one of the above commands within a program to shift the window, assign the command to a function key (with a SET KEY statement).
If you wish to specify it with a SET CONTROL statement, this statement must be followed by a REINPUT statement (that is, it must be placed between the REINPUT statement and corresponding INPUT statement); otherwise Natural will not be able to uniquely identify the window to which the command is to be applied (and will ignore it). As a rule, however, no SET CONTROL ’W’ statement should be placed between an INPUT statement with WINDOW=’window-name’ option and the corresponding REINPUT statement.

Examples of Command Combinations

The various parameters to be specified with the %W command may also be combined with one another; for example:

 Copyright © Software AG 2002

%W<<--	Position window to top left corner of page.
%W>>++	Position window to bottom right corner of page.
%W++-	Display the next to last window segment of the page.
%W+3>6	Position window 3 lines down and 6 positions right on the page.
%W10+>	Position window to line 10 of the page, then 1 window down and 1 window right.
%WL40C10++-3	Define the window with a line size of 40 positions and a page size of 10 lines, move that window to the bottom of the page, then move it up 3 lines on the page.
%WL30C10B3/15--<<	Define the window with a line size of 30 positions, page size of 10 lines, position the window at line 3 column 15 on the physical screen and move this window to the top left corner of the page.
%WFS	Generate a frame around the window and set the STAY option on.

Note:
When you specify multiple parameters with the %W command, note that on mainframe computers the maximum number of characters after the "%" is 24; any further characters will be ignored.
The parameters are evaluated in the sequence in which they are specified, so that different sequences of the same parameters may lead to different results.

%WA and %WZ - Save Screen Image before Window

With "%WA", you activate a "save screen image before window" feature. When this feature is activated and a window is to be opened, all active screen data that will be overlaid by the window are saved. When the same window is moved, the saved screen image is reconstructed before the window is built up at the new location on the screen. In addition, it is possible to rebuild the saved images of multiple dependent windows whenever the calling window becomes active again.

When the current INPUT statement uses a window, the screen image is stored before the window is output. Whenever the same INPUT statement is repeated, the current or all subsequent stored screen images are recovered and are written back to the screen.
(On mainframe computers, the screen images are stored in the WSIZE buffer, whose size can be set with the WSIZE profile parameter (see the section Profile Parameters in the Parameter Reference documentation).

This feature makes it possible, for example, to use windows in a PC-like manner. For a given window, any number of dependent windows can be written to the screen. All these windows will disappear from the screen when the main input window is re-executed.

The buffer contents (screen images) are deleted whenever Natural performs a full-screen I/O, when Natural returns to command mode (NEXT), after a LOGON command, or after the CLEAR key has been pressed.

With "%WZ", youdeactivate a previously entered "%WA" command.

%X - Control of Statistics Line/Infoline

```
%X [ [I|S] [+|-] ]
   [  B
      T
      nn
      -nn  ]
```

This terminal command controls the display of the Natural statistics line/infoline.

On mainframe computers, the line can be used as either statistics line or infoline (but not both at the same time); on all other platforms it can only be used as infoline.

%X+	switches the display of the statistics line/infoline on.
%X-	switches the display of the statistics line/infoline off.
%X	switches the display of the statistics line/infoline on and off (toggle switch).
%XI+	displays the line in infoline mode.
%XI-	displays the line in statistics display mode (only possible on mainframe computers).
%XI	switches between the two display modes (only possible on mainframe computers).
%XS+	displays additional statistics (see below).
%XS-	returns to display of original statistics.
%XS	switches between additional and original statistics.
%XB	displays the statistics line/infoline in the bottom line of the screen.
%XT	displays the statistics line/infoline in the top line of the screen.
%X*nn*	displays the statistics line/infoline on line *nn* of the screen. If the line number *nn* is not within the current screen, the statistics line/infoline will not be displayed.
%X-*nn*	displays the statistics line/infoline on the *nn*th line from the bottom of the screen. If the line number -*nn* is not within the current screen, the statistics line/infoline will not be displayed.

Infoline

Data can be written to the infoline by specifying the output destination "INFOLINE" with the DEFINE PRINTER statement. Only a single line can be written to the infoline. The infoline can be used to have status information displayed, for example, for debugging purposes; it can also be used as separator line (as defined by SAA standards).

Statistics Line - on Mainframe Computers only

When the line is used in statistics display mode, the following statistical information is provided:

 Copyright © Software AG 2002

IO	Number of bytes transmitted to the screen during the previous screen operation.
AIO	Average number of bytes transmitted per terminal screen operation (since beginning of Natural session).
L	Logical line number of top line within the logical page shown in the current window.
C	Logical column number of the leftmost column within the logical page shown in the current window.
LS	Logical line size of current page (as defined by session parameters).
PS	Logical page size of current page (as defined by session parameters).
PLS	Physical line size of window.
PCS	Physical page size of window.
FLD	Number of fields generated in the last screen.
CLS	Number of program calls during the last terminal I/O.
ADA	Number of Adabas calls during the last terminal I/O.
Additional statistics displayed with %XS+:	
MIN	Minimum duration (in seconds) of Natural activities between two terminal I/Os in the current Natural session.
MAX	Maximum duration (in seconds) of Natural activities between two terminal I/Os in the current Natural session.
AVR	Average duration (in seconds) of Natural activities between two terminal I/Os in the current Natural session.
LST	Duration (in seconds) of Natural activities between the last but one and the last but two terminal I/Os.

%Y - Control of PF-Key Lines

```
      ( N                                  )
      ( S                                  )
      ( P                                  )
      ( W                                  )
      ( U                                  )
      ( A                                  )
      ( H                                  )
      ( B                                  )
%Y    ( T                                  )
      ( [-]nn                              )
      ( F                                  )
      ( L                                  )
      ( X                                  )
      ( I                                  )
      ( V                                  )
      ( C                                  )
      ( = color-code [color-code] [color-code] )
```

The terminal command %Y is used to control the display of the Natural PF-key lines.

Note:
On graphical user interfaces, this command is ignored.

Display Format of Function-Key Lines

%YN	displays the function-key lines in normal tabular Software AG format.
%YS	displays the function-key lines in sequential format, and only showing those keys to which names have been assigned (PF1=value,PF2=value,etc.).
%YP	displays the function-key lines in PC-like sequential format; corresponds to %YS, except that "Fn=" instead of PFn=" is displayed before the names.
%YW	corresponds to %YP - but applies only if the function-key lines are displayed within a window.
%YU	cancels the effect of %YW.

Single- and Double-Line Display

 Copyright © Software AG 2002

%YA	(all-line display) displays both function-key lines.
%YH	(half display) displays only one function-key line; for normal tabular display mode (%YN), this is the line with the function-key names; for the other two display modes (%YS and %YP), this is the upper line to be displayed.

Positioning of Function-Key Lines

%YB	displays the function-key lines at the bottom of the screen.
%YT	displays the function-key lines at the top of the screen.
%Y*nn*	displays the function-key lines on line *nn* on the screen.
%Y-*nn*	displays the function-key lines on the *nn*th line from the bottom of the screen. If the line number *nn* or *-nn* is not within the current screen, the function-key lines will not be displayed.

Range of Displayed Function Keys

%YF	displays the first range of function keys (that is, usually 1 to 12).
%YL	displays the last range of function keys (that is, usually 13 to 24).
%YX	is used to toggle-switch between the two displays.

Intensified or Reverse Video Display of Function-Key Lines

%YI	displays the function-key lines intensified. Enter "%YI" again to switch back from intensified to non-intensified display.
%YV	displays the function-key lines in reverse video. Enter "%YV" again to switch back from reverse video display to normal display.

Coloring of Function-Key Lines

%Y=*color-code(s)*	displays the function-key lines in the specified colors. The *color-codes* you can specify are the same as with the session parameter CD). You can specify up to three *color-codes*: the first *color-code* refers to the first function-key line (the one displaying the function-key numbers), the second *color-code* refers to the second function-key line (the one displaying the function-key names), and the third *color-code* refers to the background of both lines. For example, %Y=GRPIYE would cause the characters in the first line to be displayed in green, the characters in the second line in pink, and the background of both lines in yellow. **Note**: On mainframe computers, the third "color-code" will be ignored.

Cursor-Sensitivity

%YC	This command makes the function-key lines cursor-sensitive. This means that they react like an action bar on a PC screen: the user just moves the cursor to the desired function-key number or name displayed and presses ENTER, and Natural will react as if the corresponding function key had been pressed. Enter "%YC" again to switch cursor-sensitivity off again (toggle switch). By using %YC in conjunction with tabular Software AG display format (%YN) and having only the function-key names displayed (%YH), you may equip your applications with very comfortable action bar processing: the user merely has to select a function name with the cursor and press ENTER, and the function will be executed.

To activate the display of the function-key lines in a program, you use the session parameter KD=ON.

 Copyright © Software AG 2002

%Z - Clear Source Area

```
%Z
```

This command can only be used with the SET CONTROL statement.

This command clears the contents of the Natural source area.

%? - Help Information for Terminal Commands

```
%?
```

This command is only available on mainframe computers.

This terminal command displays help information on the Natural terminal commands.

%% and %. - Interrupt Current Operation

```
%%
```

Note:
The terminal commands "%%" and "%." will be ignored if the profile parameter ESCAPE = OFF (which is described in your Natural Operations documentation) is set.

%% in Online Mode

If you enter "%%" in any field on the screen, the currently active Natural program will be terminated immediately and Natural will return to command input mode.
If you enter "%%" in command input mode, the Natural session will terminated (equivalent to the system command FIN).

"%%" has the following effects:

- The contents of the Natural stack will be deleted.
- Any logical database transaction currently being processed is backed out.
- The source program currently in the work area of the editor will not be affected.

%% in Batch Mode - on Mainframe Computers only

In batch mode, "%%" may be used to set restart points in the input files and thus ensure the synchronisation of the input files in the case of an error.

If the profile parameter CC=ON (see the section Profile Parameters in the Parameter Reference documentation) is set and an error occurs during the compilation/execution of a Natural program in batch mode, the input data stream for the SYNIN and OBJIN input files will be flushed until a line containing "%%" in the first two

positions is encountered (if no "%%" is encountered, it will be flushed until the end-of-file is reached).
Moreover, the contents of the Natural stack will be deleted.
If more data are available in the input stream, Natural resumes processing with the line after "%%".

If CC=OFF, any "%%" in the input data will be ignored.

%. in Online and Batch Mode

```
%.
```

Online, "%." is the same as "%%", except that the Natural stack is not deleted.

On mainframe computer in batch mode, "%." causes reading of input values for the current INPUT statement to be terminated.

 Copyright © Software AG 2002

%.P - Delete First Entry from Stack

```
%.P
```

This command is only available on mainframe computers.

This command deletes the topmost entry from the Natural stack.

For further information on the stack, see the Natural Programming Guide.

%.S - Read Stack Entry Without Deleting it

```
%.S
```

This command is only available on mainframe computers.

Normally, commands/data are deleted from the stack as soon as they have been read from the stack.

This command causes the next INPUT statement to read the topmost entry from the stack without deleting it. The entry is treated as input data, no matter whether it actually is data or a command.

This allows you check a stack entry and then, depending on its content, decide whether to process it or not.

To check the contents of the stack, you can also use the system variable *DATA. For further information on the stack, see the Natural Programming Guide.

%+ and %- - Natural Connection

```
% { + [N] [O] }
  { -         }
```

These commands are not available under Windows 98 and Windows NT/2000. They only apply if Natural Connection is installed.

These terminal commands are used to activate/de-activate Natural Connection.

%+	This command causes the Natural system variable *DEVICE to be set to "PC". This permits usage of the Natural Connection system.
%+N	Same as %+. In addition, this command causes no field names to be sent when uploading/downloading data. **This command is only available on mainframe computers.**
%+O	Same as %+. In addition, this command causes the new features of Natural Version 2.3 not to be supported. Use Option "O" to avoid conflicts when working with older versions of Entire Connection. **This command is only available on mainframe computers.**
%+NO **%+ON**	Combines the effects of %+N and %+O. **These commands are only available on mainframe computers.**
%-	This command causes the Natural system variable *DEVICE to be set to the setting in effect prior to execution of a %+ command.

See the Natural Connection documentation for further information.

%* - Inhibit Character Display

```
% *
```

This command may be used when entering sensitive data (e.g., passwords). %* causes all fields on the current screen to be non-displayable.

When used with the SET CONTROL statement, %* causes all fields on the next screen to be non-displayable.

%* in Batch Mode - on Mainframe Computers only

In batch mode, %* causes printing of the next input data record to be suppressed. The input line after the line which contains %* will not be printed. This is useful, for example, for suppressing the printing of passwords.

 Copyright © Software AG 2002

Example of %* in Batch Logon to Natural Security:

```
//CMSYNIN DD * %* SYSSEC,DBA,DBA ...
```

The printing of all input data in batch may be controlled with the ECHO profile parameter (see the section Profile Parameters in the Parameter Reference documentation).

%/ - End-of-File

```
%/
```

This terminal command is only applicable on mainframe computers in batch mode.

When entered in the first two positions of an input record being read with an INPUT statement in batch mode, this command causes an end-of-file condition.

%= - Assigning Colors to Fields

```
                 F
                 N
% = [ [ { O } attribute }
        { M }
        { T }
        {                } = color ]
        {   BG           }
        {   WN           }
        {   PT           }
        {   FR           } ]...2 ]
```

With this command, you can assign colors to field attributes for programs that were originally not written for color support. The command causes all fields/text defined with the specified attributes to be displayed in the specified color.

If predefined color assignments are not suitable for your terminal type, you can use this command to override the original assignments with new ones.

You can also use this command within Natural editors, for example to define color assignments dynamically during map creation.

 Copyright © Software AG 2002

General settings:	
blank	Clear color translate table.
F	Newly defined colors are to override colors assigned by program.
N	Color attributes assigned by program are not to be modified.
Field types:	
O	Output fields (AD=O).
M	Input fields; that is, input-only fields (AD=A) and modifiable fields (AD=M).
T	Text constants.
Possible field attributes:	
B	Blinking
C	Italic
D	Default
I	Intensified
U	Underlined
V	Reverse video
A color can also be assigned to the following parts of a screen:	
BG	Background
WN	Foreground (that is, fields for which no color is defined)
PT	Default page title
FR	Frame of a window
Possible colors:	
BL	Blue
GR	Green
NE	Neutral
PI	Pink
RE	Red
TU	Turquoise
YE	Yellow

Example:

%=TI=RE,OB=YE

This example assigns color red to intensified text fields and color yellow to all blinking output fields.

%<TECH - Display Technical Information

```
% < TECH
```

This terminal command corresponds with the system command TECH (see the section Profile Parameters in the Parameter Reference documentation).

%<TEST - Invoke Debugging Facility

```
% < TEST
```

This command is only available on mainframe computers.

With this terminal command, you invoke the debugging facility. It corresponds with the system command TEST (described in the Natural Reference documentation).

For details on the debugging facility, see Natural Debugger in the Debugging and Monitoring section.

CLEAR Key - Interrupt Current Operation

Pressing the CLEAR key has the same effect as the terminal command "%%".

On mainframe computers,you can also use the CLEAR key to interrupt a recording that is being re-run in film mode. Further information on recording is provided in Recording in the section Debugging and Monitoring.

Note:
Under OpenVMS and UNIX, you can use the ESC key instead of the CLEAR key - unless another function is assigned to ESC in SAGtermcap.

CTRL+D Keys - Interrupt Current Operation

This function is only available on mainframe terminals.

Pressing the keys CTRL+D has the same effect as the terminal command "%%".

RESET+ENTER Keys - Attention Interrupt

This function is only available under Com-plete on terminals connected via a local controller.

Pressing RESET and then ENTER terminates the current processing loop. The loop must contain a database access statement.

 Copyright © Software AG 2002

Keywords and Reserved Words

This document contains a list of all keywords and words that are reserved in the Natural programming language. To avoid possible naming conflicts, you are strongly recommended not to use these keywords or reserved words or components thereof as names for your data or procedures.

The statement keywords affected by the Natural profile parameter KC (Keyword Checking) or the KCHECK option of the NTCMPO macro or the COMPOPT system command (as described in the Natural Reference documentation) are shown in boldprint and italics.

.
<
<>
<=
+
+V
*
**
*CONVID
*COUNTER
*CURS-COL
*CURS-LINE
*CURSOR
*DAT4D
*DAT4E
*DAT4I
*DAT4J
*DAT4U
*DATD
*DATE
*DATG
*DATI
*DATJ
*DATN
*DATU
*DATX
*DEVICE
*DIALOG-ID
*ERROR

*ERROR-LINE
*ERROR-NR
*ERROR-TA
*EVENT
*HARDCOPY
*HARDWARE
*IN
*INIT-ID
*INIT-PROGRAM
*INIT-USER
*ISN
*LIBRARY-ID
*LINE-COUNT
*LINESIZE
*MACHINE-CLASS
*NUMBER
*OCC
*OCCURRENCE
*OI
*OPSYS
*OS
*OSVERS
*OUT
*OUTIN
*PAGE-NUMBER
*PAGESIZE
*PF-KEY
*PROGRAM
*ROWCOUNT
*TIMD
*TIME
*TIMESTMP
*TIMN
*TIMX
*TPSYS
*UI

 Copyright © Software AG 2002

*WINDOW-LS
*WINDOW-POS
*WINDOW-PS
*WINMGR
*WINMGRVERS
^<
^>
^=
-
/
>
>=
?
:=
=
A
A-AVER
A-MAX
A-MIN
A-NAVER
A-NCOUNT
A-NMIN
A-SUM
ABS
ABSOLUTE
ACCEPT
ACTION
AD
ADD
ADHOC
AFTER
AL
ALARM
ALL
AND
AND TRANSLATE

ANY
APPLIC-ID
APPLIC-NAME
AS
ASC
ASCENDING
ASSIGN
ASSIGNING
AT
AT BREAK
AT END
AT START
AT TOP
ATN
AUTO
AVER
AVG
BACKOUT
BASE
BATCH
BEFORE
BETWEEN
BLOCK
BLOCKE
BLOCKED
BOT
BOTTOM
BREAK
BROWSE
BUT
BUT NOT
BX
BY
C
CABINET
CALL

 Copyright © Software AG 2002

CALLING
CALLNAT
CAP
CAPTIONED
CAT
CATALL
CATALOG
CATLG
CC
CD
CDID
CF
CHAR
CHECK
CHILD
CIPH
CIPHER
CLASS
CLEAR
CLOSE CONVERSATION
CLOSE LOOP
CLOSE PC
CLOSE PRINTER
CLOSE WORK
CLR
CMS
COM
COMMAND
COMMIT
COMPOSE
COMPRESS
COMPUTE
CONDITION
CONST
CONSTANT
CONTEXT

CONTROL
CONVERSATION
COPIES
COPY
COS
COUNT
COUPLED
CR
CREATE
CURRENT
CURS-FIELD
CURSOR
CV
DATA
DATAAREA
DATE
DAY
DAYS
DC
DEBUG
DECIDE
DECIMAL
DEFINE
DEFINE DATA
DEFINE SERVER
DEFINE VIEW
DELETE
DELIMITED
DELIMITER
DELIMITERS
DESC
DESCENDING
DEST
DESTINATION
DIALOG
DIALOG-ID

 Copyright © Software AG 2002

DIGITS
DISP
DISPLAY
DISPLAY FORMATTED
DISTINCT
DIVIDE
DLOGOFF
DLOGON
DNATIVE
DO
DOCUMENT
DOEND
DOWNLOAD
DRAW
DU
DUMP
DY
DYNAMIC
E
EDIT
EDITED
EDT
EJ
EJECT
ELSE
EM
END
END-ACTION
END-ALL
END-BEFORE
END-BLOCK
END-BREAK
END-BROWSE
END-CLASS
END-DECIDE
END-DEFINE

END-ENDDATA
END-ENDFILE
END-ENDPAGE
END-ERROR
END-FILE
END-FIND
END-FOR
END-HISTOGRAM
END-IF
END-INTERFACE
END-JOIN
END-LOOP
END-METHOD
END-NOREC
END-PARAMETERS
END-PROCESS
END-PROPERTY
END-READ
END-REPEAT
END-SELECT
END-SERVER
END-SORT
END-START
END-SUBROUTINE
END-TOPPAGE
END-UNITE
END-VALUE
END-VALUES
END-VIEW
END-WORK
ENDHOC
ENDING
ENDING AT
ENTER
ENTIRE
ENTR

 Copyright © Software AG 2002

EQ
EQUAL TO
ERROR
ERROR-LINE
ERROR-TA
ERRORS
ES
ESCAPE
ETID
EVEN
EVENT
EVERY
EX
EXAMINE
EXEC
EXECUTE
EXISTS
EXIT
EXP
EXPORT
EXTRACTING
F
FALSE
FC
FETCH
FIELD
FIELDS
FILE
FILES
FILL
FILLER
FIN
FINAL
FIND
FIRST
FL

FLOAT
FOR
FORM
FORMAT
FORMATTED
FORMATTING
FORMS
FOUND
FRAC
FRAMED
FROM
FS
FULL
FUNCTIONS
G
GC
GDA
GE
GEN
GENERATED
GET
GFID
GIVE
GIVING
GLOBAL
GLOBALS
GRAPHICS
GREATER EQUAL
GREATER THAN
GROUP
GROUP BY
GT
H
HAVING
HC
HD

 Copyright © Software AG 2002

HE
HEADER
HELLO
HELP
HEX
HISTOGRAM
HORIZ
HORIZONTALLY
HOUR
HW
IA
IC
ID
IDENTICAL
IF
IGNORE
IM
IMMEDIATE
IMPORT
IN
INCCONT
INCDIC
INCDIR
INCLUDE
INCLUDED
INCMAC
INDEPENDENT
INDEX
INDEXED
INDICATOR
INDX
INIT
INITIAL
INPL
INPUT
INSERT INTO

INT
INTEGER
INTERCEPTED
INTERFACE
INTERMEDIATE
INTO
INVERTED
INVESTIGATE
IO
IP
IS
ISN
ISPF
JOIN
JUST
JUSTIFIED
KD
KEY
KEYS
L
LANGUAGE
LAST
LC
LE
LEAVE
LEAVING
LEFT
LENGTH
LESS
LESS EQUAL
LESS THAN
LEVEL
LIKE
LIMIT
LINDICATOR
LINES

 Copyright © Software AG 2002

LIST
LISTED
LOCAL
LOG
LOG-LS
LOG-PS
LOGICAL
LOGOFF
LOGON
LOOP
LOWER
LOWER CASE
LS
LT
M
MACROAREA
MAIL
MAINMENU
MAP
MARK
MASK
MAX
MC
MCG
MESSAGES
METHOD
MGID
MICRO
MICROSECOND
MIN
MINUTE
MIX
MODIFIED
MODULES
MONTH
MORE

MOVE
MOVING
MP
MS
MT
MULTIPLY
NAME
NAMED
NAVER
NC
NCOUNT
NE
NET
NEWPAGE
NL
NMIN
NO
NO ERASE
NO PARAMETER
NO PARMS
NODE
NOHDR
NONE
NOT
NOT <
NOT >
NOT =
NOT EQ
NOT GT
NOT LT
NOT EQUAL
NOTIT
NOTITLE
NPC
NPLCMD1
NPLCMD2

 Copyright © Software AG 2002

NPLCMD3
NULL
NULL-HANDLE
NUMBER
NUMERIC
O
OBJECT
OBTAIN
OF
OFF
OFFSET
OLD
ON
ON ACTION
ON ERROR
ONCE
OPEN
OPEN CONVERSATION
OPTIONS
OR
OR =
OR EQ
OR EQUAL
OR EQUAL TO
OR=
ORDER BY
OUTPUT
OVFLW
PAGE
PARAMETER
PARAMETERS
PARENT
PASSW
PASSWORD
PATTERN
PA1

PA2
PA3
PC
PD
PEN
PERFORM
PF-NAME
PF*n* (*n* = 1 to 9)
PF*nn* (*nn* = 10 to 99)
PGM
PHYSICAL
PLOT
PM
POS
POSITION
PR
PREV
PREVIOUS
PRIMARY
PRINT
PRINTER
PRIORITY
PROCESS
PROCESSING
PROFILE
PROGRAM
PROGRAMS
PROPERTY
PS
PURGE
QUARTER
R
READ
READONLY
RECORD
RECORDS

 Copyright © Software AG 2002

REDEFINE
REINPUT
REJECT
RELEASE
REMAINDER
RENAME
RENUM
RENUMBER
REPEAT
REPEATED
REPLACE
REPORT
REQUEST
REQUIRED
RESET
RESETTING
RESPONSE
RESTORE
RESULT
RET
RETAIN
RETAINED
RETRY
RETURN
REVERSED
RIGHT
ROLLBACK
ROUNDED
ROUTINE
RULEVAR
RUN
RUNMODE
SA
SAME
SAVE
SCAN

SCR
SCRATCH
SCREEN
SEARCH
SECOND
SELECT
SELECTION
SEND
SEPARATE
SEQUENCE
SERVER
SET
SET TIME
SETS
SETTIME
SETUP
SF
SG
SGN
SHORT
SHOW
SIN
SINGLE
SIZE
SKIP
SL
SM
SOME
SORT
SORTED
SOUND
SOURCE
SPACE
SPECIFIED
SQLID
SQRT

 Copyright © Software AG 2002

STACK
START
STARTING
STARTING FROM
STARTUP
STATEMENT
STATUS
STEP
STEPLIB
STOP
STORE
STOW
SUBPROGRAM
SUBPROGRAMS
SUBROUTINE
SUBSTR
SUBSTRING
SUBTRACT
SUM
SUPPRESS
SUPPRESSED
SUSPEND
SYMBOL
SYSTEM
T
TAN
TC
TECH
TERMINATE
TEST
TEXT
TEXTAREA
TEXTVARIABLE
THAN
THEM
THEN

THRU
TIME
TIME-OUT
TIMES
TIMESTAMP
TIMEZONE
TITLE
TO
TO VARIABLE
TO VARIABLES
TOP
TOTAL
TP
TR
TRAILER
TRANSACTION
TRANSFER
TRANSLATE
TREQ
TRUE
TS
TSO
TYPE
U
UC
UNCAT
UNCATALOG
UNCATLG
UNDERLINED
UNDLIN
UNION
UNIQUE
UNITE
UNTIL
UPDATE
UPLOAD

 Copyright © Software AG 2002

UPPER
UPPER CASE
USED
USER
USER-NAME
USING
VAL
VALUE
VALUES
VARGRAPHIC
VARIABLE
VARIABLES
VERIFY
VERSIONS
VERT
VERTICALLY
VIA
VIEW
WASTE PAPER
WH
WHEN
WHERE
WHILE
WINDOW
WITH
WORK
WRITE
X
XREF
YEAR
ZD
ZP