# Natural

## Utilities

Version 8.2.7

October 2018

ADABAS & NATURAL

Dieses Dokument gilt für Natural ab Version 8.2.7.

Hierin enthaltene Beschreibungen unterliegen Änderungen und Ergänzungen, die in nachfolgenden Release Notes oder Neuausgaben bekanntgegeben werden.

Copyright © 1979-2018 Software AG, Darmstadt, Deutschland und/oder Software AG USA, Inc., Reston, VA, USA, und/oder ihre Tochtergesellschaften und/oder ihre Lizenzgeber.

Der Name Software AG und die Namen der Software AG Produkte sind Marken der Software AG und/oder Software AG USA Inc., einer ihrer Tochtergesellschaften oder ihrer Lizenzgeber. Namen anderer Gesellschaften oder Produkte können Marken ihrer jeweiligen Schutzrechtsinhaber sein.

Nähere Informationen zu den Patenten und Marken der Software AG und ihrer Tochtergesellschaften befinden sich unter http://documentation.softwareag.com/legal/.

Diese Software kann Teile von Software-Produkten Dritter enthalten. Urheberrechtshinweise, Lizenzbestimmungen sowie zusätzliche Rechte und Einschränkungen dieser Drittprodukte können dem Abschnitt "License Texts, Copyright Notices and Disclaimers of Third Party Products" entnommen werden. Diese Dokumente enthalten den von den betreffenden Lizenzgebern oder den Lizenzen wörtlich vorgegebenen Wortlaut und werden daher in der jeweiligen Ursprungssprache wiedergegeben. Für einzelne, spezifische Lizenzbeschränkungen von Drittprodukten siehe PART E der Legal Notices, abrufbar unter dem Abschnitt "License Terms and Conditions for Use of Software AG Products / Copyrights and Trademark Notices of Software AG Products". Diese Dokumente sind Teil der Produktdokumentation, die unter http://softwareag.com/licenses oder im Verzeichnis der lizenzierten Produkte zu finden ist.

Die Nutzung dieser Software unterliegt den Lizenzbedingungen der Software AG. Diese Bedingungen sind Bestandteil der Produktdokumentation und befinden sich unter http://softwareag.com/licenses und/oder im Wurzelverzeichnis des lizenzierten Produkts.

**Dokument-ID: NATMF-NNATUTILITIES-827-20180523**

## Table of Contents


Preface ..... xvii
I Utility Activation ..... 1
1 Utility Activation ..... 3
II Utilities Grouped by Purpose ..... 5
2 Utilities Grouped by Purpose ..... 7
III ADACALL Utility - Issuing Adabas Direct Calls ..... 9
3 ADACALL Utility - Issuing Adabas Direct Calls ..... 11
Invoking ADACALL ..... 12
ADACALL Parameters ..... 13
ADACALL Commands and PF Keys ..... 15
User Exit ADAEXIT ..... 17
IV DBLOG Utility - Logging Database Calls ..... 19
4 Executing DBLOG ..... 21
Basic Principles of Database Logging ..... 22
Data Processing and Storage ..... 23
Activating and Deactivating DBLOG ..... 24
Using Selective DBLOG ..... 25
5 DBLOG Menu ..... 27
DBLOG Menu Functions ..... 28
Specifying Logging Restrictions ..... 30
Specifying Adabas Buffers ..... 30
6 DBLOG Trace Screen ..... 33
DBLOG Trace Screen for Adabas Commands ..... 34
DBLOG Trace Screen for DL/I Calls ..... 41
DBLOG Trace Screen for SQL Statements ..... 43
7 DBLOG Snapshot Function ..... 47
Snapshot Function for Adabas Commands ..... 48
Snapshot Function for DL/I Calls ..... 50
Snapshot Function for SQL Statements ..... 52
8 TEST DBLOG Command ..... 55
Syntax Diagrams ..... 56
Keyword Explanations ..... 57
V INPL Utility ..... 59
9 INPL Utility ..... 61
Introducing the INPL Utility ..... 62
Load Libraries Only ..... 68
Load DDMs Only ..... 68
Load Error Messages Only ..... 69
Load All Objects ..... 69
Replace Product Installation ..... 70
Scan INPL File ..... 71
Natural Security Recover ..... 71
User Exit Routines ..... 72

VI NATPAGE Utility - Screen Capturing ..... 75
10 NATPAGE Utility - Screen Capturing ..... 77
VII NATRJE Utility - Natural Remote Job Entry ..... 79
11 NATRJE Utility - Natural Remote Job Entry ..... 81
General Information on NATRJE ..... 82
Calling NATRJE from a Natural Program ..... 83
NATRJE Return Codes ..... 88
NATRJE Features Applicable to openUTM/TIAM ..... 89
VIII Object Handler ..... 93
12 General Information on the Object Handler ..... 95
Principles of Object Transfer ..... 96
Invoking the Object Handler ..... 97
Batch or Direct Command Calls ..... 99
Issuing Object Handler Commands from a Natural Program ..... 100
Text Members for Reports, Restarts and Traces ..... 100
Natural Security ..... 101
Standard PF Keys ..... 101
13 Functions ..... 103
14 Wizards ..... 105
Step 1 - Start the Procedure ..... 106
Step 2 - Unload/Load/Scan Objects into/from Work Files ..... 107
Step 3 - Set Parameters ..... 109
Step 4 - Select Objects ..... 110
Step 5 - Execute Processing ..... 111
Step 6 - Continue Processing ..... 112
15 Advanced User ..... 113
Activating Advanced User ..... 114
Processing Objects ..... 114
16 Compact Mode ..... 117
How to Select Compact Mode ..... 118
How Instructions are Processed in Compact Mode ..... 118
17 Restart Load ..... 121
18 View ..... 123
Natural Library Objects ..... 124
Natural System Error Messages ..... 125
Natural Command Processor Sources ..... 126
FDTs ..... 127
Natural-Related Objects ..... 127
DDMs ..... 128
19 Find ..... 131
20 Administration ..... 133
List the Available Workplans in the Workplan Library ..... 134
Create a New Workplan ..... 136
Change the Workplan Library ..... 138
Change the Report Library ..... 139

21 Select System File ..... 141
22 Select Library ..... 143
23 Select System Error Messages ..... 145
Columns and Commands ..... 146
24 Select Objects ..... 149
Columns and Commands on List Screens ..... 150
25 Object Specification ..... 153
26 Object Specification - All Objects on the Work File ..... 155
27 Object Specification - Natural Library Objects ..... 157
Natural Library Objects ..... 158
Natural Library Object Details ..... 159
Natural Library Object Properties ..... 161
Natural Library Object Exceptions ..... 162
Natural Library Object Exception Properties ..... 163
28 Object Specification - Natural System Error Messages ..... 165
Natural System Error Messages ..... 166
Natural System Error Message Details ..... 166
Natural System Error Message Exceptions ..... 167
29 Object Specification - Natural Command Processors ..... 169
Natural Command Processors ..... 170
Natural Command Processor Source Exceptions ..... 171
30 Object Specification - Natural-Related Objects ..... 173
Natural Profiles ..... 174
Natural Debug Environments ..... 175
Natural DL/I Subfiles ..... 176
31 Object Specification - DDMs ..... 179
DDMs ..... 180
DDM Properties ..... 181
DDM Exceptions ..... 182
32 Object Specification - FDTs ..... 183
33 Use Selection or List Workplan ..... 185
34 Settings ..... 187
Settings Screen Fields ..... 188
Set Additional Options ..... 190
Set Global Parameters ..... 198
35 Workplans ..... 203
Creating, Selecting and Modifying Workplans ..... 204
Contents of Workplans ..... 204
Examples of Workplans ..... 205
Referencing Workplans ..... 206
36 Name, Date and Time Specification ..... 209
Name ..... 210
Date ..... 211
Time ..... 212
37 Work Files ..... 213

Work File Assignment ... 214
Work File Format ... 214
38 Direct Commands ... 217
39 Basic Command Syntax ... 219
40 select-clause ... 223
Syntax of select-clause ... 224
SELECTION or LIST Workplan ... 224
Natural Library Object and DDM Selection ... 225
Natural-Related Debug Environment Selection ... 231
Natural-Related Profile Selection ... 232
Natural-Related DL/I Subfile Selection ... 234
Natural System Error Message Selection ... 236
Natural Command Processor Selection ... 237
FDT Selection ... 239
Application Selection ... 240
Object Selection for Delete Instructions ... 243
Help Text Selection ... 245
41 Object List - LIST Workplan ... 247
Syntax of object-type-and-location ... 248
Syntax of object-name-description ... 250
Example of an Object List ... 252
42 parameter-setting ... 253
Syntax of parameter-clause ... 254
Keyword Explanation of parameter-clause ... 255
43 option-setting ... 259
Syntax of option-setting ... 260
Keyword Explanation of option-setting ... 262
44 Examples of Using Direct Commands ... 269
Unloading Objects for the Same Platform ... 270
Unloading Objects for Different Platforms ... 271
Loading Objects in Internal Format ... 271
Loading Objects in Transfer Format ... 272
45 Commands for Navigation and Special Functions ... 273
46 Batch Condition Codes and User Exit Routines ... 277
Condition Codes Returned in Batch ... 278
Applying User Exit Routines ... 278
User Exit Routines Available ... 279
47 Tools ... 281
Status ... 282
Last Result ... 282
Traces ... 282
Reports ... 283
48 Profile Settings ... 285
PF Keys ... 287
Line Commands ... 287

Profile Parameters ... 288
49 Migration from NATUNLD/NATLOAD and SYSTRANS to the Object Handler ... 293
Converting Individual Commands ... 294
Processing Commands with a User Exit Routine ... 296
Processing SYSTRANS Commands with OBJHAPI ... 296
Unsupported SYSTRANS Options ... 297
IX Recording Utility ... 299
50 Recording Utility ... 301
Purpose of Recording ... 302
Data and Functions Recorded ... 302
Recording a Session ... 303
Playing Back a Recording ... 304
Manipulating a Recording ... 306
X SYSAPI Utility - APIs of Natural Add-On Products ... 307
51 SYSAPI Utility - APIs of Natural Add-On Products ... 309
Prerequisites ... 310
Invoking and Terminating SYSAPI ... 310
Reserved Keywords ... 311
Using the SYSAPI Utility ... 311
XI SYSBPM Utility - Buffer Pool Management ... 315
52 Invoking and Operating SYSBPM ... 317
Invoking and Terminating SYSBPM ... 318
Online Help ... 320
SYSBPM Main Menu - Fields, Functions and Commands ... 321
SYSBPM in a z/OS Parallel Sysplex Environment ... 324
53 List Objects ... 325
List Natural Objects ... 326
List Messages ... 338
54 Delete Objects ... 343
Delete Objects from Buffer Pool and/or BP Cache ... 344
Delete Objects from Message Pool ... 344
55 Directory Information ... 345
Fields for Buffer Pool Objects ... 346
Fields for BP Cache Objects ... 348
PF Keys and Direct Commands ... 349
56 Hexadecimal Display ... 351
PF Keys and Direct Commands ... 352
57 Write to Work File ... 353
58 Display Sorted Extract ... 355
59 Buffer Pool Statistics ... 357
General Buffer Pool Statistics ... 358
Buffer Pool Load/Locate Statistics ... 361
Buffer Pool Fragmentation ... 365
Internal Function Usage ... 367

Buffer Pool Hash Table Statistics ..... 367
Performance Hints ..... 373
PF Keys and Direct Commands ..... 378
60 BP Cache Statistics ..... 379
General BP Cache Statistics ..... 380
BP Cache Call Statistics ..... 382
BP Cache Hash Table Statistics ..... 384
Performance Hints ..... 385
PF Keys and Direct Commands ..... 385
61 Message Pool Statistics ..... 387
62 Select Buffer Pool ..... 389
Invoking Select Buffer Pool ..... 390
Displaying Buffer Pools ..... 391
Resetting a Buffer Pool ..... 391
63 Select Message Pool ..... 393
Invoking Select Message Pool ..... 394
Displaying Message Pools ..... 395
Resetting a Message Pool ..... 395
64 Blacklist Maintenance ..... 397
Maintain Blacklist ..... 398
List Object Sets ..... 402
Edit Object Set ..... 402
Add Object Set to Blacklist ..... 405
Delete Object Set from Blacklist ..... 406
Delete Object Set Source Object ..... 407
Additional Object Set Maintenance with Utilities ..... 407
Blacklist Maintenance in Batch Mode ..... 408
65 Preload List Maintenance ..... 411
List Preload Lists ..... 412
Edit Preload List ..... 413
Generate Preload List ..... 417
Delete Preload List ..... 419
Additional Maintenance Functions with Utilities ..... 420
66 Performance Considerations ..... 421
Internal Fast Locate Table ..... 422
Searching in Steplibs ..... 423
Reusing and Retaining Objects ..... 423
Local versus Global Buffer Pool ..... 424
67 SYSBPM Direct Commands ..... 425
68 Batch Processing ..... 433
Related Topics ..... 434
69 Application Programming Interfaces ..... 435
XII SYSCP Utility - Code Page Administration ..... 437
70 SYSCP Utility - Code Page Administration ..... 439
Invoking and Terminating SYSCP ..... 440

Code Page Maintenance ..... 442
All Code Pages ..... 454
Unicode Properties ..... 460
ICU Information ..... 461
XIII SYSEDT Utility - Editor Buffer Pool Administration ..... 463
71 SYSEDT Utility - Editor Buffer Pool Administration ..... 465
Defining a Natural Security Library Profile ..... 466
Invoking SYSEDT and Executing a Function ..... 466
General Information ..... 468
Generation Parameters ..... 469
Users ..... 470
Logical Files ..... 471
Recovery Files ..... 472
Administration Facilities ..... 472
Help on Direct Commands and Menu Functions ..... 473
XIV SYSERR Utility ..... 475
72 General Information on Messages ..... 477
Message Types ..... 479
Message Languages ..... 479
Messages and Code Page Support ..... 480
Issuing Messages ..... 480
Retrieving Natural System Short Messages ..... 481
Retrieving User-Defined Short Messages ..... 481
Obtaining Message Information ..... 482
73 Invoking SYSERR ..... 483
74 Functions ..... 485
Adding Messages ..... 486
Deleting Messages ..... 490
Displaying Messages ..... 490
Modifying Messages ..... 492
Printing Messages ..... 494
Scanning Messages ..... 496
Selecting Messages from a List ..... 498
Translating Messages into other Languages ..... 501
75 Parameters ..... 505
Message Type ..... 506
Library ..... 506
Message Number ..... 506
Language Codes ..... 506
76 Direct Commands ..... 507
77 Upper Case Conversion - ERRUPPER ..... 509
78 Replacing Characters - ERRCHAR ..... 511
79 Managing Messages in Different Libraries ..... 513
Unloading Messages - ERRULDUS ..... 514
Loading Messages - ERRLODUS ..... 516

80 Application Programming Interface USR0020P ..... 517
XV SYSEXT Utility - Natural Application Programming Interfaces ..... 519
81 SYSEXT Utility - Natural Application Programming Interfaces ..... 521
Introduction to SYSEXT ..... 522
Invoking and Terminating SYSEXT ..... 524
Using the SYSEXT Utility ..... 526
Interface Versions ..... 531
Reserved Keywords ..... 531
Using a Natural API ..... 532
XVI SYSEXV Utility ..... 535
82 SYSEXV Utility ..... 537
Executing Example Programs of Current Versions ..... 538
Executing Example Programs of Non-current Versions ..... 538
Terminating the SYSEXV Utility ..... 539
XVII SYSMAIN Utility - Object Maintenance ..... 541
83 General Information on SYSMAIN ..... 543
Basic SYSMAIN Functionality ..... 544
Object Types and Storage Location ..... 545
Overview of Functions ..... 545
84 Invoking and Terminating SYSMAIN ..... 547
Invoking SYSMAIN Online or Batch ..... 548
Invoking SYSMAIN with Appl. Programming Interface ..... 549
Terminating SYSMAIN ..... 550
85 Using Menu Functions and Commands ..... 551
Performing Menu Functions ..... 552
Executing Commands ..... 554
Description of Functions ..... 556
Function Processing and Reporting ..... 563
SYSMAIN Online Help ..... 568
86 Processing Programming Objects ..... 571
Fields in Programming Objects Menus ..... 572
Using Profile Parameter RECAT ..... 575
Selection Lists for Programming Objects ..... 575
XRef Considerations ..... 579
Specifying Additional Criteria ..... 581
Direct Command Syntax for Programming Objects ..... 583
87 Processing Debug Environments ..... 587
Fields in the Debug Environments Menu ..... 588
Selection Lists for Debug Environments ..... 590
Direct Command Syntax for Debug Environments ..... 591
88 Processing Error Messages ..... 595
Fields in Error Message Menus ..... 596
Selection Lists for Error Messages ..... 598
Renumbering Error Messages ..... 599
Specifying Languages ..... 600

Direct Command Syntax for Error Messages ..... 601
89 Processing Profiles ..... 605
Fields in the Profiles Menu ..... 606
Selection Lists for Profiles ..... 608
Direct Command Syntax for Profiles ..... 609
90 Processing Rules ..... 613
Fields in the Rules Menu ..... 614
Selection Lists for Rules ..... 615
Direct Command Syntax for Rules ..... 617
91 Processing DL/I Subfiles ..... 621
Fields in the DL/I Subfiles Menu ..... 622
Selection Lists for DL/I Subfiles ..... 623
Direct Command Syntax for DL/I Subfiles ..... 624
92 Processing DDMs ..... 627
Fields in the DDMs Menu ..... 628
Selection Lists for DDMs ..... 630
Direct Command Syntax for DDMs ..... 631
93 Processing Predict Sets ..... 635
Fields in the Predict Sets Menu ..... 636
Selection Lists for Predict Sets ..... 637
Direct Command Syntax for Predict Sets ..... 638
94 Keywords and Variables in Direct Commands ..... 641
Description of Keywords ..... 642
Specifying a Range of Names ..... 653
95 Special Commands Issued to SYSMAIN ..... 655
96 Processing Status and Error Notification ..... 659
Object Rejection and Reasons ..... 660
Status Messages ..... 661
SYSMAIN Error Notification ..... 665
97 Special Considerations for Administrators ..... 669
File Security ..... 670
Natural Security ..... 671
User Exit Routines ..... 672
XVIII SYSNCP Utility ..... 679
98 SYSNCP Utility ..... 681
Introducing the SYSNCP Utility ..... 682
Invoking SYSNCP ..... 688
Processor Selection ..... 689
Header Records ..... 690
Keyword Maintenance ..... 700
Function Maintenance ..... 705
Runtime Actions ..... 710
Processor Cataloging ..... 715
Administrator Services ..... 715
Session Profile ..... 723

XIX SYSPARM Utility ..... 727
99 SYSPARM Utility ..... 729
Invoking SYSPARM ..... 730
List Profiles ..... 731
Display Profile ..... 733
Add New Profile ..... 734
Modify Profile ..... 734
Editing Profiles ..... 735
Copy Profile ..... 739
Delete Profile ..... 739
Direct Commands and Batch Processing ..... 740
Maintaining Profiles in Different Environments ..... 748
Invoking Help on Parameters from the Command Line ..... 748
XX ..... 751
100 Natural Profiler ..... 753
101 Profiling Natural Applications ..... 755
Introducing Profiling ..... 756
Platform-Specific Profiling ..... 756
Profiling Tools ..... 757
Natural Profiler Evaluations ..... 759
102 Code Coverage of Natural Applications ..... 767
Introducing Code Coverage ..... 768
Platform-Specific Code Coverage ..... 769
Code Coverage Tools ..... 769
Natural Code Coverage Evaluations ..... 772
103 Basic Concepts of the Profiler Utility ..... 779
Profiler Utility Overview ..... 780
Data Collection in Batch ..... 784
Data Consolidation, Code Coverage and Data Processing ..... 785
Sampling ..... 789
Profiler Performance in Batch ..... 791
Profiling Long-Running Applications ..... 792
Related Topics ..... 796
104 Using the Profiler Utility in Online Mode ..... 797
Prerequisites ..... 798
Invoking and Terminating the Profiler Utility Online ..... 798
Events ..... 799
Functions ..... 800
105 Using the Profiler Utility in Batch Mode ..... 811
Quick Start for Profiling ..... 812
Quick Start for Code Coverage ..... 815
Prerequisites ..... 818
Invoking and Terminating the Profiler Utility ..... 821
Syntax and Keywords ..... 821
Events and Data Collected ..... 825

Initializing Profiling ... 830
Initializing Code Coverage ... 833
Starting and Pausing Data Collection ... 836
Using Filters to Limit the Data Collected ... 839
Enabling Sampling ... 844
Writing User-Defined Events ... 845
Monitor Session CMPRMIN ... 846
Profiling a Batch Natural RPC Server ... 848
Profiling a Mainframe Session from Natural Studio ... 849
Consolidating Event Data ... 851
Evaluating Event Data ... 853
Exporting Event Data for MashZone ... 871
Maintaining Profiler Resource Files ... 872
Including Profiler Input from Natural Text Objects ... 876
Event Trace ... 877
Tracing Natural Code Coverage ... 879
Internal Trace ... 882
Profiler Statistics ... 884
106 Natural Profiler MashApp ... 893
Preparing to Use the MashApps ... 894
Preparing the Profiler Data ... 898
Opening the MashApps ... 899
Evaluation Page ... 900
Compare Page ... 909
Properties Page ... 911
Use Cases ... 913
XXI SYSRDC Utility ... 931
107 SYSRDC Utility ... 933
Functional Components of SYSRDC ... 934
Data-Collecting Events ... 935
Data Collected ... 937
Activating and Controlling the Natural Data Collector ... 939
Trace Recording ... 940
User Exits for External Monitoring/Accounting ... 941
Calling the CMRDC Interface ... 942
Example Programs ... 946
XXII SYSRPC Utility ... 951
108 Invoking and Terminating SYSRPC ... 953
Invoking SYSRPC ... 954
Terminating SYSRPC ... 955
Invoking Online Help ... 955
109 Service Directory Maintenance ... 957
Service Directory Concept ... 958
Invoking Service Directory Maintenance ... 959
Fields on the Service Directory Screen ... 961

Commands for Service Directory Maintenance ..... 964
110 Replacing Items in the Service Directory ..... 969
Syntax of SYSRPC SM REPLACE ..... 970
111 Generating Interface Objects - General Considerations ..... 973
112 Generating Single Interface Objects with Parameter Specification ..... 975
Using the Interface Object Generation Function ..... 976
Specifying Parameters ..... 979
Examples of Interface Object Generation ..... 981
113 Generating Multiple Interface Objects ..... 985
Using the SYSRPC SGMASS Direct Command ..... 986
Name Specification and Compression ..... 988
114 Calculating Size Requirements ..... 991
Using the SYSRPC CSMASS Direct Command ..... 992
Name Specification and Compression ..... 994
115 Parameter Maintenance ..... 995
Invoking Parameter Maintenance ..... 996
Specifying NTRPC/RPC Keyword Subparameters ..... 996
116 Server Command Execution ..... 999
Using Server Command Execution ..... 1000
Pinging an RPC Server ..... 1002
Terminating an RPC Server ..... 1004
117 Listing Servers Registered on EntireX Broker ..... 1007
Example of an SYSRPC SRVLIST Direct Command ..... 1009
Viewing a Server List ..... 1009
Viewing Additional Server Information ..... 1010
Customizing Server Lists ..... 1011
118 Overview of SYSRPC Direct and Batch Commands ..... 1013
XXIII SYSTP Utility ..... 1015
119 Invoking SYSTP and Executing Functions ..... 1017
120 Using SYSTP Utility Screens ..... 1019
121 General SYSTP Functions ..... 1021
Natural Monitoring (SYSMON) ..... 1022
Natural Print/Work Files (SYSFILE) ..... 1026
Natural Swap Information ..... 1027
Buffer Usage Statistics (BUS) ..... 1031
Natural Sub-Systems and Roll Server Information ..... 1033
Natural Thread Usage Statistics ..... 1034
Natural License Information ..... 1037
122 SYSTP Functions under CICS ..... 1039
Natural User Sessions ..... 1040
Natural Roll Facilities ..... 1045
Natural Thread Groups ..... 1046
Natural Storage Threads ..... 1047
NCI Global System Information ..... 1048
NCI Generation Options ..... 1050

Natural Thread Group Definitions ..... 1051
Own Natural User Session ..... 1052
CICS Task Information ..... 1052
System Administration Facilities ..... 1052
123 SYSTP Functions under IMS TM ..... 1055
Broadcasting ..... 1056
Display Environment Data ..... 1056
Monitoring ..... 1057
Applied NII Zaps ..... 1057
124 SYSTP Functions under TIAM and UTM ..... 1059
P-Key Utility ..... 1060
Show Common Memory Pools ..... 1064
125 SYSTP in Batch for CICS Sessions ..... 1065
Invoking SYSTP in Batch Mode ..... 1066
Evaluating the Log File ..... 1066
Index ..... 1069

# Preface

This document describes the purpose and use of the utilities provided by Natural.

| | |
|---|---|
| **Utilities Grouped by Purpose** | Lists all utilities grouped according to their purpose. |
| **Utility Activation** | Describes how Natural invokes a utility. |
| **ADACALL** | Issues Adabas direct calls (native commands) directly to an Adabas database. |
| **DBLOG** | Logs database calls: indicates which Adabas commands, DL/I calls or SQL statements are issued by a Natural program. |
| **INPL** | Loads or scans Natural objects supplied by Software AG. |
| **NATPAGE Screen Capturing** | Captures screens (maps and reports) during a Natural session. |
| **NATRJE** | Submits JCL cards from a Natural program to the operating system for scheduling and execution. |
| **Natural Profiler** | Monitors the internal process flow of a Natural application and analyzes the performance and code coverage of the application. |
| **Object Handler** | Processes Natural and non-Natural objects for distribution in Natural environments. This is done by unloading the objects in the source environment into work files and loading them from work files into the target environment. |
| **Recording** | Records commands and input data entered during a Natural session. Re-executes a recorded session. |
| **SYSAPI** | Locates Application Programming Interfaces (APIs) provided by Natural add-on products. |
| **SYSBPM** | Monitors and controls the Natural buffer pool. |
| **SYSCP** | Provides code page information and can be used to administrate code pages for Natural source objects. |
| **SYSEDT** | Displays parameters and runtime information for the editor buffer pool. Modifies parameters and deletes logical work and recovery files. |
| **SYSERR** | Creates application-specific messages. In addition, it can be used to modify the texts of the existing Natural system messages (not recommended). |
| **SYSEXT** | Locates Natural Application Programming Interfaces (APIs). |
| **SYSEXV** | Provides examples of the new features of the current Natural versions. |
| **SYSMAIN** | Performs object operations in Natural such as copy, move and delete. |
| **SYSNCP** | Defines command-driven navigation systems for Natural applications. |
| **SYSPARM** | Creates and maintains Natural parameter profiles. |
| **SYSRDC** | Enables a Natural application to record monitoring and accounting data on the processing flow. |
| **SYSRPC** | Establishes and maintains Natural RPC (Remote Procedure Call) environments. |

| | |
|---|---|
| **SYSTP** | Monitors and controls TP-monitor-specific characteristics of Natural. |

See also in the *Editors* documentation.

# I Utility Activation

# 1 Utility Activation

Natural invokes a Natural utility without performing a logon to the corresponding utility library in the FNAT system file. As a result, Natural preserves the global data area (GDA) and/or application-independent variables (AIV). The current user library and the settings are maintained. (To reset the GDA and/or the AIVs, see the profile parameter `FREEGDA` in the *Parameter Reference*.)

To preserve the settings of your application environment, do *not* log on to a utility library. Instead, invoke a utility by using the Natural system command that corresponds to the utility.

After terminating a utility, you will be returned to the library from which you invoked the utility. However, if you explicitly log on to a utility library before invoking the utility, you will stay in this (utility) library after utility termination.

**Exception:**

The utilities SYSEXT and SYSEXV still perform an implicit logon to the corresponding utility library since object sources can only be edited within an active library.

For information on how to control the use of Natural utilities with Natural Security, see the section *Protecting Utilities* in the *Natural Security* documentation.

If Natural Security is not installed, you can control the use of Natural utilities with user exit routine UTI-EX01. The program source for this user exit routine is provided as source object UTI-SX01 in library SYSEXT.

### To activate UTI-EX01

1 `CATALOG` or `STOW` source object UTI-SX01 under the name UTI-EX01.

   Different names are used to guarantee that the source object (possibly modified according to your requirements) and the cataloged object of the user exit routine are not overwritten by an update installation.

2 Copy UTI-EX01 to library SYSTEM in the FNAT or the FUSER system file.

For a detailed description of the user exit routine, see the source object of UTI-SX01 in the library SYSEXT.

# II Utilities Grouped by Purpose

# 2 Utilities Grouped by Purpose

The following is a list of all Natural utilities grouped according to their purpose:

| Administration | Debugging | Monitoring | Object Transfer |
|---|---|---|---|
| SYSAPI | ADACALL | SYSBPM | INPL |
| SYSCP | DBLOG | SYSEDT | Object Handler |
| NATRJE | Debugger | SYSRDC | SYSMAIN |
| SYSBPM | DUMP | SYSTP | |
| SYSEDT | NATPAGE Screen Capturing | | |
| SYSERR | Recording | | |
| SYSEXT | SYSRDC | | |
| SYSNCP | | | |
| SYSPARM | | | |
| SYSRPC | | | |
| SYSTP | | | |

# III ADACALL Utility - Issuing Adabas Direct Calls

# 3 ADACALL Utility - Issuing Adabas Direct Calls


- Invoking ADACALL ..... 12
- ADACALL Parameters ..... 13
- ADACALL Commands and PF Keys ..... 15
- User Exit ADAEXIT ..... 17

The utility ADACALL can be used to issue Adabas direct calls (native commands) to an Adabas database for learning and testing and for analyzing problems.

The utility ADACALL is contained in the library SYSADA.

## Invoking ADACALL

### To invoke ADACALL

- Enter the following system command:

```
SYSADA
```

An ADACALL main screen similar to the example screen below is displayed:

```
15:53:32            ***** NATURAL ADACALL UTILITY *****            2006-12-14
 User SAG                 - ADABAS Direct Calls -
 Mode Char                                                   Call No. 45
 *** Control Block ***         First Byte 30
  Cmd L3        Cmd ID SAG           File 316             Database 10
 Resp 0            ISN 382            ISQ 0                    ISL 0
  FBL 210          RBL 980            SBL 140     VBL 140      IBL 0
 COP1             COP2          User Area                 Cmd Time 4
 Addition1        Addition2  Addition3        Addition4        Addition5
 AA]?             227  48
 *** Buffer Areas ***
 Format  AA,AC,AE.

 Record  11111003ARTHUR               DENT

 Search

  Value

    ISN
Command ===>
Enter-PF1---PF2---PF3---PF4---PF5---PF6---PF7---PF8---PF9---PF10--PF11--PF12---
      Help  Main  Exit              Char  Hex   View  Prnt  Run   Init  Canc
```

On the ADACALL main screen, specify the necessary parameter values and execute the Adabas command by either choosing PF10 (Run) or entering the ADACALL command `EXEC` in the Command line.

In the example screen above, the Adabas command `L3` was executed for a logical read of the employees file.

Except for the control block, which is shown in full, only a part of the buffer is displayed. You can view the buffers in their entirety by using any of the ADACALL direct commands or PF keys listed below.

## ADACALL Parameters

The parameters which can be specified on the ADACALL main screen are listed below. You can use the ADACALL online help function to obtain a summarized explanation of the parameters.

### To invoke the online help function

- Place the cursor in the field for which you require help and enter a question mark (?) or choose PF1. (For read-only fields, only PF1 applies.)

For detailed information, see the Adabas documentation *Command Reference* and *Messages and Codes*.

| Parameter | Explanation |
|---|---|
| **Mode** | Indicates the display mode of the buffer contents:<br>`Char` — Character values.<br>`Hex` — Hexadecimal values.<br>To change modes, see the ADACALL commands `CHAR` and `HEX`. |
| **Call No.** | Number of commands executed since the start of the session. |
| **First Byte** | The first byte of the Adabas control block.<br><br>Indicates whether 1-byte or 2-byte database IDs (DBID) and file numbers (FNR) are used:<br><br>H'00' = 1-byte DBID, FNR (file numbers 1 - 255)<br>H'30' = 2-byte DBID, FNR (file numbers greater than 255) |
| **Cmd** | Adabas command.<br><br>Enter and execute the **Adabas `OP` command** to specify the parameters described in the relevant section below. |
| **Cmd ID** | Command ID. |
| **File** | File number.<br><br>If **First Byte** is set to `H'00'`:<br>3-digit file number, **Database** not equal to 0. |

| Parameter | Explanation |
|---|---|
| | If **First Byte** is set to `H'30'`:<br>5-digit file number. |
| **Database** | Database ID (DBID). Defaults to the DBID of the FUSER file of the current Natural session (see **File** above).<br><br>If **First Byte** is set to `H'30'`, then the database number will be moved to the response code field of the Adabas control block at execution time. |
| **Resp** | Response code returned after the command is executed. |
| **ISN** | Internal sequence number. |
| **ISQ** | ISN quantity. |
| **ISL** | Lowest ISN value for ISN lists. |
| **FBL** | Format buffer length in bytes (maximum 210). |
| **RBL** | Record buffer length in bytes (maximum 980). |
| **SBL** | Search buffer length in bytes (maximum 140). |
| **VBL** | Value buffer length in bytes (maximum 140). |
| **IBL** | ISN buffer length in bytes (maximum 200). |
| **COP1** | Command option 1. |
| **COP2** | Command option 2. |
| **User Area** | User area for the control block. |
| **Cmd Time** | The time taken to execute the command, converted to 1/100th seconds for convenience. |
| **Addition1** | Additions 1. |
| **Addition2** | Additions 2. If the call was successful, it displays the compressed length of the record being read and the decompressed length of the data requested via the format buffer. If a non-zero response is returned and the error was a result of an invalid format buffer, the field in error and its offset into the format buffer are displayed. |
| **Addition3** | Additions 3. |
| **Addition4** | Additions 4. If a VSAM file is being read, this field is set to VSAM if initialized. |
| **Addition5** | Additions 5. |
| **Format** | Format buffer. (The final period is necessary.) |
| **Record** | Record buffer. |
| **Search** | Search buffer. (The final period is necessary.) |
| **Value** | Value buffer. |
| **ISN** | ISN buffer. |

### Adabas OP Command

When you execute the Adabas command OP (Open), ADACALL provides a window where you can specify the following parameters:

- maximum ISNs to be stored in the internal ISN buffer,
- maximum records permitted in hold status,
- maximum CIDs (command IDs) which may be active,
- maximum time permitted for execution of an Sx command.

In the window, enter the relevant information and choose ENTER.

For an explanation of the parameters and valid values, refer to the *Adabas Command Reference* documentation.

## ADACALL Commands and PF Keys

The ADACALL direct commands listed below are provided to change ADACALL parameter settings or to switch between screens by either entering a command in the Command line or choosing a corresponding PF key.

In addition to ADACALL commands, from the Command line, you can also issue Natural system commands.

In the following table, an underlined portion of a command represents an acceptable abbreviation.

| Command | PF Key | Function |
|---|---|---|
| | PF1 | Invoke the help function for ADACALL. If the cursor is positioned on one of the various ADACALL parameters and PF1 is pressed, help information on this parameter is displayed. |
| | PF2 | Return to the ADACALL main screen. **Mode** is set to CHAR. |
| BACK | PF5 | Page backward to the previous buffer when viewing the buffers in their entirety.<br><br>Valid only after the VIEW command has been applied, which means that the command is not applicable from the ADACALL main screen. |
| CB | | Display the control block buffer entirely; valid in hexadecimal mode only. |
| CHAR | PF6 | Change the current mode to character mode (EBCDIC). |
| D | | Display extended error message text for response code received. When an Adabas response other than 0 (zero) is returned, the corresponding short error message text is displayed in the message line. The extended text can be viewed by issuing this command. |
| EXEC<br><br>or | PF10 | Execute the direct command with the parameters specified. |

| Command | PF Key | Function |
|---|---|---|
| RUN | | |
| EXIT<br>or<br>STOP<br>or<br>Q<br>or<br>. | PF3<br>or<br>PF12 | Exit. If pressed while on the ADACALL main screen, ADACALL is terminated. If one of the buffer screens is being viewed, the ADACALL main screen is displayed with **Mode** unchanged. |
| FB | | Display the format buffer in its entirety. |
| FWD | PF4 | Page forward to the next buffer when viewing the buffers in their entirety.<br><br>Valid only after the VIEW command has been applied, which means that the command is not applicable from the ADACALL main screen. |
| HEX | PF7 | Change the current mode to hexadecimal. |
| IB | | Display the ISN buffer in its entirety. |
| INIT | PF11 | Initialize/reset buffer(s). A window is displayed and one of the following values can be entered for the buffers indicated:<br><br>H — Initialize the corresponding buffer(s) with binary zeroes (H'00').<br><br>any character except H or blank — Initialize the corresponding buffer(s) with blanks (H'40').<br><br>blank character — Do not initialize the corresponding buffer(s).<br><br>If you enter INIT ALL, all buffers except the control block are initialized with blanks. Alternatively, the command INIT FB RB SB VB IB (not all buffers need be listed) can be specified and all buffers in the list are initialized with blanks.<br><br>**Note:** The ISN buffer is always initialized with binary zeroes. |
| PRINT | PF9 | Generate and display a report on the status of all buffers.<br><br>The Natural terminal command %H can be used to obtain a hardcopy. |
| RB | | Display the record buffer in its entirety. |
| RUN | | Same as EXEC. |
| SB | | Display the search buffer in its entirety. |

| Command | PF Key | Function |
|---|---|---|
| VB | | Display the value buffer in its entirety. |
| VIEW | PF8 | Display all buffers in their entirety. The first buffer to be displayed is the record buffer. The FWD command can be used to page through the other buffers.<br><br>If you VIEW the record buffer in hexadecimal mode, the data are displayed on four pages:<br><br>To page forwards, enter the command FWD or choose PF4.<br>To page backwards, enter the command BACK or choose PF5.<br>To display a specific page, enter a page number from 1 to 4 in the field **Specify next page number**.<br><br>To view buffers individually, enter any of the following commands:<br><br>FB — Format buffer<br>RB — Record buffer<br>SB — Search buffer<br>VB — Value buffer<br>IB — ISN buffer<br>CB — Control block (default). Valid in hexadecimal mode only: change to HEX before executing VIEW. |
| VSAM | | If VSAM has been defined for the current Natural session, this direct command can be issued to access or update VSAM files. When you issue this command, you are prompted by a window for the VSAM file name. When the command is executed, it is directed to the appropriate VSAM file. |

## User Exit ADAEXIT

ADACALL allows direct commands to be issued to any database. Therefore, as a means of security, a user exit is supplied. This user exit is called ADAEXIT and is contained in the library SYSADA. You can modify ADAEXIT as required. The Adabas control block is passed as a parameter to ADAEXIT. You can change the source code of the user exit so as to modify the contents of the control block. By simply changing the database ID or file number, or by setting the Command Code to XX, you can prevent database calls from being performed.

# IV DBLOG Utility - Logging Database Calls

The DBLOG utility is used to log Adabas commands, or DL/I and SYNC/ROLB calls, or SQL statements. Logging is useful for tuning an application (controlling the flow of commands accessing the database) and for analyzing error codes that may be returned from the database.


**Executing DBLOG**

**DBLOG Menu**

**DBLOG Trace Screen**

**DBLOG Snapshot Function**

**TEST DBLOG Command**

# 4 Executing DBLOG

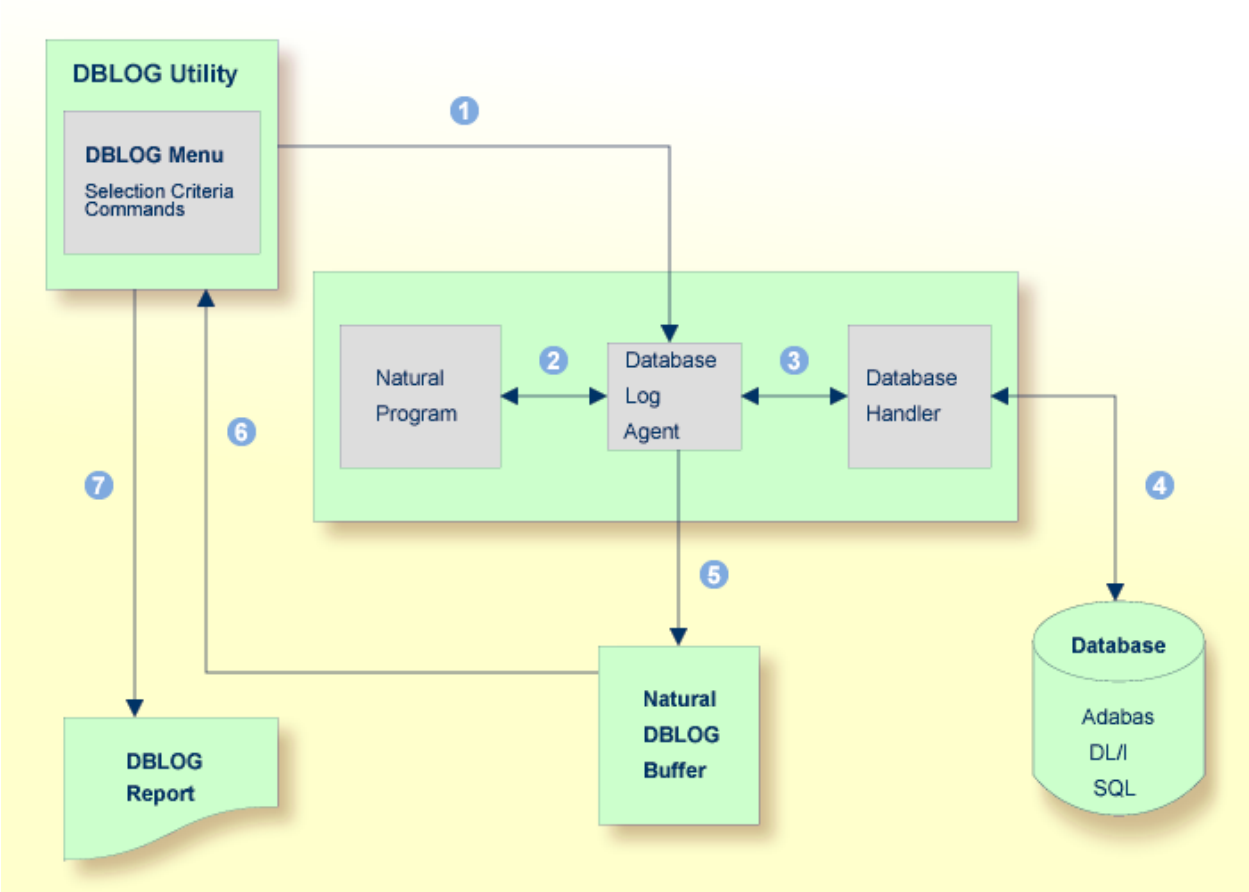

- Basic Principles of Database Logging ..... 22
- Data Processing and Storage ..... 23
- Activating and Deactivating DBLOG ..... 24
- Using Selective DBLOG ..... 25

The DBLOG utility logs each Adabas command, or DL/I and SYNC/ROLB call, or SQL statement after it has been processed by the database system. Logging starts when you activate DBLOG and execute or run a Natural program.

This section covers the following topics:

## Basic Principles of Database Logging

The following graphic illustrates the process flow when database calls are being logged from a Natural program with the DBLOG utility:



### Legend

1. The logging of database calls (Adabas commands, or DL/I and SYNC/ROLB calls, or SQL statements) is activated by using the corresponding **DBLOG Menu** function or the Natural system command `TEST DBLOG`.
   In the **DBLOG Menu**, you can specify individual selection criteria (for example, restrict the logging to a particular database ID or file number).
2. A Natural program issues a statement that initiates a database call, for example, `FIND` or `READ`.
3. The database log agent forwards the database call to the database handler.
4. The database handler adapts the database call to the particular database (Adabas, DL/I or SQL), retrieves the data requested by the database call and returns this data to the database log agent.
5. The database log agent records in the Natural DBLOG buffer the data returned by the database handler and forwards this data to the Natural program.
6. The report function of the DBLOG utility reads the data recorded in the Natural DBLOG buffer and selects records according to the selection criteria specified in the **DBLOG Menu** in 1.
7. From the data records selected, the report function of the DBLOG utility generates a report that can be displayed, written to a work file or used for batch processing.

## Data Processing and Storage

The data logged by the DBLOG utility is recorded in the Natural DBLOG buffer. The initial and maximum sizes of the buffer are determined by the `DSIZE` profile parameter described in the *Parameter Reference* documentation.

If there is not enough space to hold the data in the DBLOG buffer, Natural increases the DBLOG buffer size appropriately (possibly several times) until the maximum size specified with the `DSIZE` profile parameter is reached. After the maximum is reached, only the most recent log data is held in the Natural DBLOG buffer.

If the DBLOG buffer size cannot be further increased due to lack of storage, Natural issues a NAT7545 message that indicates insufficient space for the DBLOG buffer. All data logged prior to this lack-of-storage situation will be kept in the Natural DBLOG buffer and can be displayed by using the `TEST DBLOG` command.

DBLOG can be used online or in batch mode. DL/I and SYNC/ROLB calls can be logged under CICS, under IMS TM or in batch mode. For further information on batch-mode processing, refer to *Natural in Batch Mode* described in the *Operations* documentation.

The logs recorded are displayed on the **DBLOG Trace** screen.

The DBLOG utility provides default settings for data recording. When using the **DBLOG Menu**, you can specify selection criteria for the commands, calls or statements to be logged and the information displayed. The **DBLOG Menu** also provides functions for activating or deactivating logging. You can also use the Natural system command `TEST DBLOG` to control DBLOG execution.

The fields of the **DBLOG Trace screen**, the **DBLOG Menu** and the Natural system command `TEST DBLOG` are explained in the relevant sections of the DBLOG documentation.

## Activating and Deactivating DBLOG

The commands used to activate or deactivate DBLOG with the default DBLOG utility settings are described in the following section. See also ***TEST DBLOG Command*** for additional information.

### To activate or deactivate DBLOG for Adabas

- Enter the following Natural system command (toggle command):

  ```
  TEST DBLOG
  ```

  Or:

  Enter the following to activate:

  ```
  TEST DBLOG ON
  ```

  Enter the following to deactivate:

  ```
  TEST DBLOG OFF
  ```

  Or:

  In the **DBLOG Menu**, enter function code `B` (to activate) or function code `E` (to deactivate).

### To activate or deactivate DBLOG for DL/I

- Enter the following Natural system command (toggle command):

  ```
  TEST DBLOG D
  ```

  Or:

  Enter the following to activate:

  ```
  TEST DBLOG D ON
  ```

  Enter the following to deactivate:

  ```
  TEST DBLOG D OFF
  ```

Or:

In the **DBLOG Menu**, enter function code `B` (to activate) or function code `E` (to deactivate).

### To activate or deactivate DBLOG for SQL

- Enter the following Natural system command (toggle command):

  ```
  TEST DBLOG Q
  ```

  Or:

  Enter the following to activate:

  ```
  TEST DBLOG Q ON
  ```

  Enter the following to deactivate:

  ```
  TEST DBLOG Q OFF
  ```

  Or:

  In the **DBLOG Menu**, enter function code `B` (to activate) or function code `E` (to deactivate).

## Using Selective DBLOG

The following are example instructions for logging Adabas commands, DL/I calls or SQL statements with selection criteria specified in the **DBLOG Menu**.

### To perform DBLOG with selection criteria

1 Invoke the **DBLOG Menu** by entering one of the following Natural system commands:

  - For Adabas:

    ```
    TEST DBLOG MENU
    ```

  - For DL/I:

    ```
    TEST DBLOG D MENU
    ```

  - For SQL:

    ```
    TEST DBLOG Q MENU
    ```

  The **DBLOG Menu** appears.

2 In the **DBLOG Menu**, specify logging restrictions and activate logging: complete the input fields and enter function code `B`.

   The message `DBLOG started now` is displayed.

3 Execute a Natural program which contains Adabas commands, DL/I calls or SQL statements.

4 Invoke the **DBLOG Trace** screen and deactivate logging by entering one of the following Natural system commands:

   - For Adabas:

     ```
     TEST DBLOG
     ```

   - For DL/I:

     ```
     TEST DBLOG D
     ```

   - For SQL:

     ```
     TEST DBLOG Q
     ```

   The **DBLOG Trace** screen appears.

5 Clear the Natural DBLOG buffer and deactivate logging by entering one of the following Natural system commands:

   - For Adabas:

     ```
     TEST DBLOG OFF
     ```

   - For DL/I:

     ```
     TEST DBLOG D OFF
     ```

   - For SQL:

     ```
     TEST DBLOG Q OFF
     ```

   DBLOG terminates and the NEXT prompt appears.

See also the section ***TEST DBLOG Command*** for additional information.

# 5 DBLOG Menu


- DBLOG Menu Functions ..... 28
- Specifying Logging Restrictions ..... 30
- Specifying Adabas Buffers ..... 30

In the **DBLOG Menu**, you can activate or deactivate logging and specify which Adabas commands, DL/I calls or SQL statements are to be logged.

### To invoke the DBLOG Menu

- Enter one of the following Natural system commands:
  - For Adabas:

    ```
    TEST DBLOG MENU
    ```

  - For DL/I:

    ```
    TEST DBLOG D MENU
    ```

  - For SQL:

    ```
    TEST DBLOG Q MENU
    ```

This section covers the following topics:

## DBLOG Menu Functions

The functions provided in the **DBLOG Menu** are described in the following section. You can execute a function by either entering the code that corresponds to the required function in the **Code** field or pressing the PF key that corresponds to the required function.

| Code or PF Key | Function | Explanation |
|---|---|---|
| B<br>or<br>PF4 | **Begin Logging of Adabas Commands** | Activates the DBLOG logging of the Adabas commands, DL/I calls or SQL statements that match the selection criteria.<br><br>See also alternative commands in *TEST DBLOG Command*.<br><br>See also **Optional Buffers for Codes B and W**. |
| E<br>or<br>PF5 | **End and Display Log Records** | Deactivates logging and displays the **DBLOG Trace** screen of the current log record if data exists in the Natural DBLOG buffer. Current log data is kept in the Natural DBLOG buffer.<br><br>See also alternative commands in *TEST DBLOG Command*. |

| Code or PF Key | Function | Explanation |
|---|---|---|
| S<br>or<br>PF6 | **Snapshot of Specific Adabas Commands** | Adabas: Interrupts a program at a specified Adabas command and displays detailed information on this command only: see *Snapshot Function for Adabas Commands*.<br><br>DL/I and SQL: Collects detailed information on a specified DL/I call or SQL statement: see Snapshot Function for *Snapshot Function for DL/I Calls* and *Snapshot Function for SQL Statements*. |
| W<br>or<br>PF7 | **Write Log Records to Work File** | Writes the logged data contained in the Natural DBLOG buffer to a work file. The file structure (columns and log sequence) corresponds to the structure of the **DBLOG Trace** screen described in *DBLOG Trace Screen.*<br><br>When you choose this function, a **Work File Selection** window prompts you for the following information:<br><br>■ Specify the target work file:<br>Enter N (No; this is the default setting) to output the data on Natural Work File 1.<br><br>Or:<br>If Entire Connection is installed, enter Y (Yes) to output the data on a PC text file by using Natural Work File 7.<br><br>■ Specify whether to write data logged for the Adabas control block to the work file:<br>Enter N (No; this is the default setting) to include the data.<br><br>Or:<br>Enter Y (Yes) to exclude the data.<br><br>See also **Optional Buffers for Codes B and W**. |
| | **Optional Buffers for Codes B and W** | Only applicable to Adabas commands.<br><br>Selects additional Adabas buffers to be logged when using function code B or W: see *Specifying Adabas Buffers*. |
| PF3 | | Exits the **DBLOG Menu**. The current log records are kept in the Natural DBLOG buffer. |
| PF12 | | Clears the Natural DBLOG buffer, leaves the **DBLOG Menu** and returns to the NEXT prompt. |

## Specifying Logging Restrictions

This section describes the input fields the **DBLOG Menu** provides for specifying selection criteria to restrict logging:

| Field | Explanation |
|---|---|
| **Skip** | Only applicable with function code S.<br><br>Number of commands, calls or statements to be skipped before logging is to start. |
| **Program** | Restricts logging to commands, calls or statements issued by the program specified. |
| **DBID** | Only applicable to Adabas commands.<br><br>Restricts logging to commands issued for the database ID specified. |
| **FNR** | Only applicable to Adabas commands.<br><br>Restricts logging to commands issued for the file number specified. |
| **Line from**<br>**Line to** | Restricts logging to commands, calls or statements within the range of the source line numbers specified. |
| **Low Resp**<br>**High Resp** | Only applicable to Adabas commands.<br><br>Restricts logging to commands which result in a response code within the range specified. |
| **Low Stat**<br>**High Stat** | Only applicable to DL/I calls.<br><br>Restricts logging to calls which result in a status code within the range specified. |
| **Low SQLC**<br>**High SQLC** | Only applicable to SQL statements.<br><br>Restricts logging to statements which result in an SQL return code within the range specified. |

## Specifying Adabas Buffers

Only applicable to Adabas commands.

The Adabas control block is logged by default. It is either the classic Adabas control block (ACB) or the extended Adabas control block (ACBX) depending on the command executed. For detailed information on Adabas control blocks, see *Adabas Control Block Structures (ACB and ACBX)* in the *Adabas* for mainframes documentation.

In addition to the control block, you can log one or more Adabas buffers listed in the **DBLOG Menu**:

| | |
|---|---|
| **FB** | Format buffer |
| **RB** | Record buffer |
| **SB** | Search buffer |
| **VB** | Value buffer |
| **IB** | ISN buffer |

You enable logging of these buffers and specify the range of bytes to be logged by using the input fields below the buffer names:

- **Bytes 0-79**

  Mark the buffer(s) to be logged by entering any character next to the required buffer(s). By default, a maximum of 80 bytes (from 0 to 79) is logged per buffer if no value is entered in the **From** and **To** fields.

- **From**

  You can enter a start number (for example, `100`) that determines from which byte the buffer is logged.

  If you want to log an entire buffer (maximum is 32 KB), enter `X` or `*` (asterisk) and leave the **To** field blank.

- **To**

  You can enter an end number (for example, `200`) that determines up to which byte the buffer is logged. The maximum input value is `32767` (32 KB).

If the length of the buffer(s) to be logged exceeds the total limit of 2097151 KB (2 GB - 1 byte), Natural issues an appropriate message as described in *Data Processing and Storage*.

The logs of the buffers can be displayed on the **DBLOG Trace** screen as described in *Displaying Adabas Buffers*.

**Note:** The snapshot function (see the relevant section) logs all Adabas buffers by default. Therefore, you need not mark any of the optional buffers before you execute this function.

# 6 DBLOG Trace Screen


- DBLOG Trace Screen for Adabas Commands ..... 34
- DBLOG Trace Screen for DL/I Calls ..... 41
- DBLOG Trace Screen for SQL Statements ..... 43

The **DBLOG Trace** screen displays recorded log data on Adabas commands, or DL/I and SYNC/ROLB calls, or SQL statements which are kept in the Natural DBLOG buffer.

This section covers the following topics:

## DBLOG Trace Screen for Adabas Commands


- Invoking DBLOG Trace for Adabas Commands
- Screen Columns and Commands on DBLOG Trace
- Displaying Adabas Buffers
- Displaying Adabas Commands that use Multi-Fetch


### Invoking DBLOG Trace for Adabas Commands

The following is an example instruction for invoking the **DBLOG Trace** screen for Adabas commands.

1. Write the following Natural program:

```
DEFINE DATA LOCAL
1 EMP-VIEW VIEW OF EMPLOYEES
  2 NAME
END-DEFINE
READ (3) EMP-VIEW BY NAME
  DISPLAY NAME
END-READ
END
```

2. Enter the following Natural system command

```
TEST DBLOG
```

   The message `DBLOG started now` is displayed.

3. Enter the following Natural system command:

```
RUN
```

   The Natural program in the source area is executed.

4. Enter again:

```
TEST DBLOG
```

   Logging is deactivated and a **DBLOG Trace** screen similar to the example below appears:

```
14:14:23              ***** NATURAL TEST UTILITIES *****              2015-11-03
User SAG                        - DBLOG Trace -                   Library SAG
M ________No Cmd ___DB __FNR _Rsp _______ISN ____ISQ CID(Hex) OP_ Pgm_____ Line
_          1  S1    10  2430                          00000000     ATEST2   5470
_          2  RC    10                                00000000 F   ATEST    0220
_          3  L3    10   316             295          00500101  A  LOGTEST  0050
_          4  L3    10   316             621          00500101  A  LOGTEST  0050
_          5  L3    10   316             715          00500101  A  LOGTEST  0050
_          6  RC    10   316                          00500101 SI  LOGTEST  0050
_          7  RC    10                                00000000 F   LOGTEST  0080


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

## Screen Columns and Commands on DBLOG Trace

This section describes the columns of fields contained in the **DBLOG Trace** screen and the commands available to scroll in the screen or in a buffer window opened from the screen (see *Displaying Adabas Buffers*). You execute a command by either pressing a PF key or entering a direct command in the Command line.

| Column | PF Key<br>Direct Command | Explanation |
|---|---|---|
| **M** | | Input option for line commands that invoke extra windows with detailed information on buffers: see *Displaying Adabas Buffers*. |
| **No** | | Sequence number. The commands are displayed in the sequence in which they were executed. |
| **Cmd** | | Adabas command. If the command is prefixed with an asterisk (e.g. ***S1**), the database call was submitted in ACBX shape. |
| **DB** | | Database ID. |
| **FNR** | | File number. |
| **Rsp** | | Adabas response code. |
| **ISN** | | Internal sequence number of record. |
| **ISQ** | | ISN quantity. |
| **CID** | | Command ID. |
| **CID (Hex)** | | Command ID in hexadecimal format. |
| **OP** | | Adabas Command Options 1 and 2. If the left option is a less-than sign (e.g. <A), the record was taken from the multi-fetch buffer instead of being provided by the database. |
| **Pgm** | | Program name. |
| **Line** | | Source code line number. |

| Column | PF Key<br>Direct Command | Explanation |
|---|---|---|
| | PF2 | Prints a hardcopy of a screen shot. |
| | PF3 | Exits the **DBLOG Trace** screen or closes a buffer window. The current log records are kept in the Natural DBLOG buffer. |
| | PF5 | Moves log entries to the top of the screen:<br>In column **M**, position the cursor next to the desired command and sequence number listed in column **No** and choose PF5.<br>The logs are repositioned starting with the sequence number selected. |
| | PF6<br>or<br>- - | Scrolls to the beginning of a list or the data in a buffer window. |
| | PF7<br>or<br>- | Scrolls up one page in a list or the data in a buffer window. |
| | PF8<br>or<br>+ | Scrolls down one page in a list or the data in a buffer window. |
| | PF9<br>or<br>++ | Scrolls to the end of a list or the data in a buffer window. |
| | PF10 | Only available in a buffer window with multiple record/format buffers.<br>Displays the previous record/format buffer. |
| | PF11 | Only available in a buffer window with multiple record/format buffers.<br>Displays the next record/format buffer. |
| | PF12 | Clears the Natural DBLOG buffer and deactivates logging. |

## Displaying Adabas Buffers

The Adabas control block is recorded by default. If you want to record one or more Adabas buffers, you need to mark the buffer(s) required in the **DBLOG Menu** before executing the logging function as described in *Specifying Adabas Buffers*. For example, if only logging of the format buffer has been marked in the **DBLOG Menu**, you can only display the **Format Buffer** window but not the **Record Buffer** window.

### To display control block or buffer information

1 In the input field next to the required command, enter the line command that corresponds to the required buffer and press ENTER:

| Line Command | Requested Buffer |
|---|---|
| C | Control block |
| F | Format buffer |
| R | Record buffer |
| S | Search buffer |
| V | Value buffer |
| I | ISN buffer |
| . | A period (.) exits the **DBLOG Trace** screen. The current log records are kept in the Natural DBLOG buffer. |

A window opens with the log data of the control block or buffer requested. If you entered several line commands, you can press PF3 to view the control block or buffer of the next command.

The following is an example of a window that contains data of a record buffer:

```
16:50:05             ***** NATURAL TEST UTILITIES *****             2015-11-03
User SAG                        - DBLOG Trace -                  Library SAG
M ________No Cmd ___DB __FNR _Rsp _______ISN ____ISQ CID(Hex) OP_ Pgm_____ Line
_          1  S1    10  2430                        00000000     ATEST2   5470
_          2  RC    10                              00000000 F   ATEST    0220
_          3  L3    10   316            295         00500101  A  LOGTEST  0050
R          4  L3    10   316            621         00500101  A  LOGTEST  0050
_ +------------------Page 1 of 1 (logged range:0x-0x4F)-------------------+ 050
_ | _____ Seq No 4          Record Buffer (length:0x14)                  | 050
_ | 0000 * C1C6C1D5 C1E2E2C9 C5E54040 40404040 * AFANASSIEV       * 0000  | 080
_ | 0010 * 40404040 00000000 00000000 00000000 *                  * 0010  | 110
_ +-----------------------------------------------------------------------+ 270


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

The fields provided in a buffer window are explained in the following table:

| Field | Explanation |
|---|---|
| **Page** | The number of the current page and the total number of pages generated for the buffer (in the example above, `1 of 1`). |
| **logged range** | The buffer length actually logged in hexadecimal format (in the example above, `0x0-0x4F`). |
| **Seq No** | The sequence number of the command. In the example above, the command was executed in the third place (`3`). |
| *`buffer-type`*<br><br>*`num-current`*<br>**/***`num-total`* | *`buffer-type`* denotes the type of buffer requested.<br><br>In addition, for a format or record buffer, the number of record or format buffers is displayed:<br><br>*`num-current`* — Denotes the number of the record/format buffer currently shown.<br><br>*`num-total`* — Denotes the total number of the record/format buffers logged.<br><br>For a database call that uses the extended Adabas control block (ACBX), multiple format/record buffers are logged. The example above shows the first record from a total of 13 records (`1 / 13`).<br><br>For detailed information on ACBX, see *Adabas Control Block Structures (ACB and ACBX)* in the *Adabas* for mainframes documentation. |
| **length** | The total length of the record in hexadecimal format (in the example above, `0x7A`). |
| _____ | In the input field next to **Seq No**, you can enter one of the following line commands:<br><br>`C` — Displays the control block.<br><br>`F` or `R` — Displays the format (`F`) or record (`R`) buffer.<br><br>If pairs of format and record buffers exist, entering `F` in a record buffer or `R` in a format buffer will display the matching record buffer or format buffer respectively. For example, if the second record buffer is currently displayed, entering `F` will invoke a window with the corresponding second format buffer.<br><br>`I` — Displays the ISN buffer.<br><br>`S` — Displays the search buffer. |

| Field | Explanation | |
|---|---|---|
| | V | Displays the value buffer. |
| | | |
| | *buffer-number* | You can enter the number of the record/format buffer you want to view.<br><br>See also Step 2 below. |
| | | |
| | . | A period (.) closes the current buffer window. |

2 In a record/format buffer window that contains multiple record/format buffers, you can use one of the following methods to view each record/format buffer:

Press PF10 to display the previous record/format buffer.

Or:

Press PF11 to display the next record/format buffer.

Or:

In the _____ input field, enter the number that corresponds to the record/format buffer you want to view.

### Displaying Adabas Commands that use Multi-Fetch

If the MULTI-FETCH clause is used in a FIND, READ or HISTOGRAM statement, only the Adabas commands that retrieve a set of records actually access the database. The records retrieved are moved into the multi-fetch buffer from where they are fetched during the execution of the database loop. The next database call is only made for the next set of records. For details, see *MULTI-FETCH Clause* in the *Programming Guide*.

The **DBLOG Trace** screen lists both database calls and non-database calls: a database call is marked with an M in the first position of the **OP** column, whereas a non-database call for the multi-fetch buffer is marked with the less-than sign (<). This is demonstrated in the following example.

**Example of an Adabas Command with Multi-Fetch**

Execute DBLOG for the following Natural program called MFETCH:

```
DEFINE DATA LOCAL
1 EMP-VIEW VIEW OF EMPLOYEES
  2 NAME
END-DEFINE
*
READ (5)  MULTI-FETCH OF 3  EMP-VIEW BY NAME = 'ADKINSON'
  DISPLAY *COUNTER NAME
END-READ
END
```

A **DBLOG Trace** screen similar to the example below appears:

```
10:04:46             ***** NATURAL TEST UTILITIES *****             2015-11-03
User SAG                       - DBLOG Trace -                  Library SAG
M ________No Cmd ___DB __FNR _Rsp _______ISN ____ISQ CID(Hex) OP_ Pgm_____ Line
_          1  S1    10  2430                         00000000     ATEST2   5470
_          2  RC    10                               00000000 F   ATEST    0220
_          3  L3    10   316             295         00600101 MA  MFETCH   0060
_          4  L3    10   316             621         00600101 <A  MFETCH   0060
_          5  L3    10   316             715         00600101 <A  MFETCH   0060
_          6  L3    10   316             535         00600101 MA  MFETCH   0060
_          7  L3    10   316            1038         00600101 <A  MFETCH   0060
_          8  RC    10   316                          00600101 SI  MFETCH   0060
_          9  RC    10                               00000000 F   MFETCH   0090


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

The `L3` commands listed as sequence numbers `1` and `4` retrieve a set of records from the database (indicated by `M` in the **OP** column) and return the first record back to the program. The remaining records are cached in the multi-fetch buffer.

The `L3` commands listed as sequence numbers `2`, `3` and `5` retrieve the record from the multi-fetch buffer (indicated by `<` in the **OP** column) and return it to the program.

**Contents of Record Buffer for Multi-Fetch Database Calls**

The record buffer of a database call that uses multi-fetch contains the data of all records retrieved from the database. They are listed in the sequence in which they are processed.

When loading a set of records, Adabas overwrites the record buffer from the first byte to the extent of the records which are returned from the database. Any space left in the buffer is not cleared but still contains data of old records loaded during a previous database call. This means, for example, that if a field defined as `NAME(A20)` is read and a multi-fetch factor of 5 is used, the record buffer has a length of 100 (20 * 5) bytes. If only 3 records are returned from the database, the record buffer is only filled properly with the first 3 records (bytes 1 to 60), whereas the last 2 records (bytes 61 to -100) remain unchanged.

# DBLOG Trace Screen for DL/I Calls


- Invoking DBLOG Trace for DL/I Calls
- Screen Columns on DBLOG Trace


## Invoking DBLOG Trace for DL/I Calls

The following are example instructions for invoking the **DBLOG Trace** screen for DL/I calls.

1. Write the following Natural program:

```
DEFINE DATA LOCAL
01 COURSE VIEW OF DNDL01-COURSE
  02 COURSEN (A3)
  02 TITLE (A33)
01 OFFERING VIEW OF DNDL01-OFFERING
  02 COURSEN-COURSE (A3)
  02 LOCATION (A31)
END-DEFINE
READ (5) COURSE BY COURSEN
  IF TITLE = 'NATURAL'
    FIND (1) OFFERING WITH COURSEN-COURSE = COURSEN
      MOVE 'DARMSTADT' TO LOCATION
      UPDATE
      END OF TRANSACTION
    END-FIND
  END-IF
END-READ
END
```

2. Enter the following Natural system command:

```
TEST DBLOG D
```

   The message `DBLOG started now` is displayed.

3. Enter the following Natural system command:

```
RUN
```

   The Natural program contained in the source area is executed.

4. Enter again:

```
TEST DBLOG D
```

   Logging is deactivated and the **DBLOG Trace** screen for DL/I screen is displayed:

```
User SAG                          - DBLOG Trace -                 Library SAG
No  Func PCB NS SC DBD/PSB    First SSA (truncated)   IOA (trunc) Program  Line
--- ---- --- -- -- -------- -------------------------- ----------- -------- ----
  1 PCB            PCNQA42                                         LOGDL1   0090
  2 GU   1   1     DNDL01   COURSE  *--(COURSEN =>     .           LOGDL1   0090
  3 GN   1   1     DNDL01   COURSE  *--(COURSEN =>     .Z01        LOGDL1   0090
  4 GN   1   1     DNDL01   COURSE  *--(COURSEN =>     .001        LOGDL1   0090
  5 GN   1   1     DNDL01   COURSE  *--(COURSEN =>     .004NATURA LOGDL1   0090
  6 GHNP 1   2     DNDL01   COURSE  *- (COURSEN  =004  ?010791DAR LOGDL1   0110
  7 REPL 1         DNDL01                              ?010791DAR LOGDL1   0130
  8 SYNC                                                           LOGDL1   0140
  9 PCB            PCNQA42                                         LOGDL1   0110
 10 GU   1   1     DNDL01   COURSE  *--(COURSEN = 004  .004NATURA LOGDL1   0110
 11 GHNP 1   2     DNDL01   COURSE  *--(COURSEN = 004  ?010791DAR LOGDL1   0110
 12 GN   1   1     DNDL01   COURSE  *--(COURSEN =>     +110        LOGDL1   0090
***** End of Log *****





NEXT                                                                LIB=SAG
```

### Screen Columns on DBLOG Trace

The columns of fields provided on the **DBLOG Trace** screen for DL/I calls are described in the following section.

| Column | Explanation |
|---|---|
| **No** | Sequence number. The calls are displayed in the sequence in which they were executed. |
| **Func** | DL/I function. |
| **PCB** | PCB number. |
| **NS** | Number of SSAs. |
| **SC** | DL/I status code. |
| **DBD/PSB** | DBD name for DB calls.<br>PSB name for scheduling calls. |
| **First SSA** | First 25 bytes of the first SSA. |
| **IOA** | First 13 bytes of the I/O area. |
| **Program** | Natural program name. |
| **Line** | Source-code line number. |

# DBLOG Trace Screen for SQL Statements


- Invoking DBLOG Trace for SQL Statements
- Screen Columns and Commands on DBLOG Trace


## Invoking DBLOG Trace for SQL Statements

The following is an example of invoking the **DBLOG Trace** screen for SQL statements.

1. Write the following Natural program:

```
DEFINE DATA LOCAL
01 EMP VIEW OF DSN8810-EMP
  02 EMPNO
  02 FIRSTNME
  02 MIDINIT
  02 LASTNAME
  02 EDLEVEL
  02 SALARY
01 EMPPROJACT VIEW OF DSN8810-EMPPROJACT
  02 EMPNO
  02 PROJNO
  02 ACTNO
  02 EMPTIME
END-DEFINE
FIND (1) EMP WITH EMPNO > '000300'
  FIND (1) EMPPROJACT WITH EMPNO = EMPNO(0150)
    MOVE 0.75 TO EMPTIME
    UPDATE
  END-FIND
  ADD 1 TO EDLEVEL
  UPDATE
END-FIND
*
FIND (1) EMP WITH EMPNO > '000300'
  FIND (1) EMPPROJACT WITH EMPNO = EMPNO(0240)
    DISPLAY EMPPROJACT EMP.EDLEVEL
  END-FIND
END-FIND
ROLLBACK
END
```

2. Enter the following Natural system command:

```
TEST DBLOG Q
```

   The message `DBLOG started now` is displayed.

3. Enter the following Natural system command:

```
RUN
```

The Natural program in the source area is executed.

4. Enter again:

```
TEST DBLOG Q
```

Logging is deactivated and a **DBLOG Trace** screen for SQL statements similar to the example below appears:

```
11:28:58              ***** NATURAL Test Utilities *****             2008-07-28
User SAG                        - DBLOG Trace -                Library SAG
M No   R SQL Statement (truncated)     CU SN SREF M Typ SQLC/W Program  Line LV
_    1   SELECT EMPNO,FIRSTNME,MIDINIT 01 01 0150 D DB2        LOGSQL   0150 01
_    2     FETCH CURSOR NEX            01 01 0150 D DB2        LOGSQL   0150 01
_    3   SELECT EMPNO,PROJNO,ACTNO,EMP 02 02 0160 D DB2        LOGSQL   0160 01
_    4     FETCH CURSOR NEX            02 02 0160 D DB2        LOGSQL   0160 01
_    5   UPDATE DSN8810.EMPPROJACT SET 02 03 0160 D DB2        LOGSQL   0180 01
_    6   CLOSE CURSOR                  02 02 0160 D DB2        LOGSQL   0160 01
_    7   UPDATE DSN8810.EMP SET EDLEVE 01 04 0150 D DB2        LOGSQL   0210 01
_    8   CLOSE CURSOR                  01 01 0150 D DB2        LOGSQL   0150 01
_    9   SELECT EMPNO,FIRSTNME,MIDINIT 05 05 0240 D DB2        LOGSQL   0240 01
_   10     FETCH CURSOR NEX            05 05 0240 D DB2        LOGSQL   0240 01
_   11   SELECT EMPNO,PROJNO,ACTNO,EMP 06 06 0250 D DB2        LOGSQL   0250 01
_   12     FETCH CURSOR NEX            06 06 0250 D DB2        LOGSQL   0250 01
_   13   CLOSE CURSOR                  06 06 0250 D DB2        LOGSQL   0250 01
_   14   CLOSE CURSOR                  05 05 0240 D DB2        LOGSQL   0240 01
_   15   ROLLBACK                      00 00 0000 D DB2        LOGSQL   0290 01
_
_
Command ===>


Enter-PF1---PF2---PF3---PF4---PF5---PF6---PF7---PF8---PF9---PF10--PF11--PF12---
      Help  Print Exit  Top   Posi  Bot    -     +                      Canc
```

### Screen Columns and Commands on DBLOG Trace

The columns of fields and commands provided on the **DBLOG Trace** screen for SQL statements are described in the following section. You execute a command by either pressing a PF key or entering a direct command in the Command line.

| Column | PF Key / Direct Command | Explanation |
|---|---|---|
| **M** | | Input option for line commands:<br><br>`E` — Executes the `EXPLAIN` command which provides information on the DB2 or SQL/DS optimizer's choice of strategy for executing SQL statements.<br><br>See also subsection *Using the EXPLAIN Command with Natural for DB2* and *Using the EXPLAIN Command with Natural for SQL/DS* both in the documentation on the *LISTSQL* command in the *System Commands* documentation.<br><br>`L` — Executes the `LISTSQL` command which lists the Natural statements in the source code of an object and the corresponding SQL statements into which they have been translated. An SQL statement is identified by the library name, program name, and line number taken from the Natural DBLOG buffer.<br><br>See also *LISTSQL Command* in the *System Commands* documentation.<br><br>**Important:** Since both commands obtain their information from the Natural system file, unwanted results may occur if the corresponding Natural program has been recataloged after the logging function was executed with the `TEST DBLOG Q` command. These unwanted results may be caused by statements modified after the logging. |
| **No** | | Sequence number; the statements are displayed in the sequence in which they were executed. |
| **R** | | Only applicable if the Natural file server for DB2 is in use.<br><br>Indicates by an asterisk in front of the corresponding statement that a reselection has been performed; if not, the column is left blank.<br><br>See also *Concept of the File Server* in the *Natural for DB2* documentation. |
| **SQL Statement** | | The first 29 characters of the logged SQL statement. |
| **CU** | | Cursor number. |
| **SN** | | Internal statement number. |
| **SREF** | | Statement reference number. |
| **M** | | Mode: `D` for dynamic or `S` for static. |

| Column | PF Key<br>Direct Command | Explanation |
|---|---|---|
| **Typ** | | Database type: DB2 or /DS. |
| **SQLC/W** | | Either the SQL return code in the SQLCODE field of the `SQLCA`, or the warning in the SQLWARN0 field of the `SQLCA` if `SQLCODE` is `0`. |
| **Pgm** | | Natural program name. |
| **Line** | | Source code line number. |
| **LV** | | Program level. |
| | PF2 | Prints a hardcopy of a screen shot. |
| | PF3 | Exits the **DBLOG Trace**. The current log records are kept in the Natural DBLOG buffer. |
| | PF4 | Scrolls to the beginning of the list. |
| | PF5 | Moves log entries to the top of the screen: In column M, position the cursor next to the desired command and sequence number listed in column No and choose PF5. The logs are repositioned starting with the sequence number selected. |
| | PF6 | Scrolls to the end of the list. |
| | PF7<br>or<br>- | Scrolls up one page in the list. |
| | PF8<br>or<br>+ | Scrolls down one page in a list. |
| | PF12 | Clears the Natural DBLOG buffer and deactivates logging. |

# 7 DBLOG Snapshot Function


- Snapshot Function for Adabas Commands ..... 48
- Snapshot Function for DL/I Calls ..... 50
- Snapshot Function for SQL Statements ..... 52

The snapshot function provides detailed information on one particular Adabas command, DL/I call or SQL statement.

This section covers the following topics:

## Snapshot Function for Adabas Commands

This snapshot function interrupts program execution after executing the first Adabas command that matches the selection criteria specified in the **DBLOG Menu**. The **Snapshot Report** (see the following example screen) generated for the specified Adabas command is displayed immediately after program interruption.

The snapshot function automatically logs *all* Adabas buffers. Therefore, you do not have to mark any of the optional buffers in the **DBLOG Menu** before you start the snapshot function. The default **Snapshot Report** displays the control block (CB), which is either the classic control block (ACB) or the extended Adabas control block (ACBX).

This section covers the following topics:

- Invoking Snapshot Report for Adabas Commands
- Displaying Buffers on Snapshot Report

### Invoking Snapshot Report for Adabas Commands

**To invoke the Snapshot Report screen for Adabas commands**

1. In the **DBLOG Menu**, specify an Adabas command and additional criteria, if desired, and enter function code `S`.

   The message `DBLOG snapshot facility started now` is displayed.

2. Execute a Natural program which contains the Adabas command specified in the **DBLOG Menu**.

   The program stops executing and a **Snapshot Report** screen similar to the example below appears:

```
16:36:39            ***** NATURAL TEST UTILITIES *****             2006-12-12
                            - Snapshot Report -

Command Code : L3           Command ID   :  ??? 00200101 File Number  : 013C
Response Code:      0       ISN          :          1300
ISN Low Limit: 00000000     ISN Quantity :             0
FB Length    : 0009         RB Length    : 0014          SB Length    : 0008
VB Length    : 0014         IB Length    : 0000          Com. Option 1:
Com. Option 2: V            Additions 1  : AE]?          Additions 2  :  ? ?
Additions 3  :              Additions 4  :
Global FID   : 0000000000000000 Command Time : 00000019 Pgm: SAGTEST  Lin: 0020
Control Block
0000 * 30D5D3F3 00200101 013C0000 00000514 * ?NL3 ?????    ?? * 0000
0010 * 00000000 00000000 00090014 00080014 *          ? ? ? ? * 0010
0020 * 000000E5 C1C5BBCA 40404040 00120014 *    VAE]?     ? ? * 0020
0030 * 00000000 00000000 00000000 00000000 *                  * 0030
0040 * 00000000 00000000 00000019 00000000 *            ?     * 0040
0050 * 00000000 00000000 00000000 00000000 *                  * 0050
0060 * 00000000 00000000 00000000 00000000 *                  * 0060
0070 * 00000000 00000000 00000000 00000000 *                  * 0070


Command ===> CB_____________________________________________________________
Enter-PF1---PF2---PF3---PF4---PF5---PF6---PF7---PF8---PF9---PF10--PF11--PF12---
      Help        Exit  CB    FB    RB     -     +    SB    VB    IB    Canc
```

### Displaying Buffers on Snapshot Report

The **Snapshot Report** screen shows the control block (CB) by default. If you want to display different Adabas buffers or scroll through a report, choose the appropriate PF key or, in the Command line, enter its equivalent direct command described in this section.

The availability of a PF key depends on the buffer currently displayed. If a buffer extends beyond the screen or contains multiple format/record buffers, PF keys required to scroll through the buffer are provided on the screen.

| PF Key | Direct Command | Buffer |
|---|---|---|
| PF4 | CB | Displays the control block. This is the default. |
| PF5 | FB | Displays the format buffer. |
| PF6 | RB<br>or<br>-- | Displays the record buffer (RB) or scrolls (--) to the beginning of a long buffer. |
| PF7 | - | Scrolls up one page in a long buffer. |
| PF8 | + | Scrolls down one page in a long buffer |

| PF Key | Direct Command | Buffer |
|---|---|---|
| PF9 | SB<br><br>or<br>++ | Displays the search buffer (SB) or scrolls (++) to the end of a long buffer. |
| PF10 | VB<br><br>or<br>< | Displays the value buffer (VB). For multiple format/record buffers, shows the previous (<) record/format buffer. |
| PF11 | IB<br><br>or<br>> | Displays the ISN buffer (IB). For multiple format/record buffers, shows the next (>) record/format buffer. |

For information on the fields displayed in a control block or buffer, see *Displaying Adabas Buffers*.

## Snapshot Function for DL/I Calls

This snapshot function generates the **Snapshot Report** (see the following example screen) of the first DL/I call that matches the selection criteria specified in the **DBLOG Menu**. A snapshot does not interrupt the program flow. The snapshot data is kept in the Natural DBLOG buffer to be displayed only if the user enters the appropriate DBLOG command as described below.

This section covers the following topics:

- Invoking Snapshot Report for DL/I Calls
- Snapshot Report Information for DL/I Calls

### Invoking Snapshot Report for DL/I Calls

**To invoke the Snapshot Report screen for DL/I calls**

1. In the **DBLOG Menu**, specify a DL/I call and additional criteria, if desired, and enter function code S.

   The message DBLOG snapshot facility started now is displayed.
2. Execute a Natural program which contains the DL/I call specified in the **DBLOG Menu**. (Log data is written to the Natural DBLOG buffer.)
3. Display the snapshot data by entering the following command:

   ```
   TEST DBLOG D
   ```

   Or:

In the **DBLOG Menu**, enter function code `E`.

A **Snapshot Report** screen for DL/I calls similar to the example below appears:

```
16:33:27                ***** NATURAL DBA Utility *****              2006-12-12
User SAG                       - Snapshot Report -              Library SAG
         DL/I Call ......... : GHNP           Pgm: SAGTEST     Line: 0110
         PSB Name .......... : PBNDL01
         PCB Number ........ : 1              out of 3
         Contents of this PCB                 Contents of NDB/NSB
         Database Name ..... : DNDL01         DBD is physical
         Segment Level ..... : 2              Segment Level .. : 2
         Statuscode ........ : BLANK
         Processing Options  : A
         Segment Name ...... : OFFERING       DBID/FNR ....... : 246   12
                                              Min/Max Length . : 41    41
         Length of KFBA .... : 9
         Number of SENSEGs . : 10  <WARNING>  Number of SENSEGs: 5
         Number of SSAs .... : 2

IOA:     Char :  ?010791DARMSTADT
         Zone : 02FFFFFFCCDDEECCE4444444444444444444444444400000000000000
         Digit: 0901079141942314300000000000000000000000000000000000000000

KFBA:    Char : 004010791
         Zone : FFFFFFFFF0000000000000000000000000000000000000000000000000
         Digit: 0040107910000000000000000000000000000000000000000000000000
MORE

SSA 1 :  Char : COURSE  *- (COURSEN  =004)
         Zone : CDEDEC445644CDEDECD447FFF500000000000000000000000000000000
         Digit: 36492500C00D364925500E004D000000000000000000000000000000

SSA 2 :  Char : OFFERING*F-
         Zone : DCCCDCDC5C640000000000000000000000000000000000000000000000
           Digit: 66659957C600000000000000000000000000000000000000000000000
```

### Snapshot Report Information for DL/I Calls

The following information is provided on the **Snapshot Report** screen for DL/I calls:

- the PSB (Program Specification Block) Name
- the PCB (Program Communication Block) Number
- the PCB mask, which consists of the following:
  - DBD (Database Description) including Database Name
  - Segment Level number
  - Statuscode

- Processing Options
- Segment Name
- Length of KFBA (Key Feedback Area)
- Number of SENSGs (Sensitive Segments)
- KFBA:
  Key Feedback Area
- Number of SSAs (Segment Search Argument)

- all SSAs
- the I/O Area

The first 120 bytes of the Key Feedback Area, of all SSAs (up to 15 SSAs are possible) and of the I/O area are displayed, both in decimal and hexadecimal format.

The DBD Name in the PCB is used to read the corresponding NDB (Natural equivalent of DBD) from the Natural FDIC system file. In this NDB, the segment whose name is given in the PCB is located and its minimum/maximum length and segment level number are displayed. The segment level number should match the number in the PCB. In this way, it is possible to detect inconsistencies between Natural NDBs and DL/I DBDs.

The PSB name is used to read the corresponding NSB (Natural equivalent of PSB) from the Natural FDIC system file. From this NSB, the number of sensitive segments is displayed. This number should match the number in the PCB. In this way, it is possible to detect inconsistencies between Natural NSBs and DL/I PSBs.

The snapshot function checks whether the DL/I DBD/PSB and the Natural NDB/NSB contain the same values in the fields **Level Number** and **Number of SENSEGs**. The same values, however, do not necessarily ensure that the DL/I DBD/PSB and the Natural NDB/NSB are fully consistent.

In the **example above**, the values in the **Number of SENSEGs** fields are different, because the Natural NATPSB procedure was not executed after the PSB had been changed by the DL/I PSBGEN procedure.

## Snapshot Function for SQL Statements

The snapshot function generates the **Snapshot Report** (see the following example screen) of the first SQL statement that matches the selection criteria specified in the **DBLOG Menu**. A snapshot does not interrupt the program flow.

Unlike the statements displayed with the DBLOG trace function, the snapshot shows the statement in its entirety (limited to 13 lines).

The snapshot data is kept in the Natural DBLOG buffer to be displayed only if the user enters the appropriate `DBLOG` command as described below.

This section covers the following topics:

- Invoking Snapshot Report for SQL Statements
- Snapshot Report Information for SQL Statements

## Invoking Snapshot Report for SQL Statements

### To invoke the Snapshot Report screen for SQL statements

1. In the **DBLOG Menu**, specify an SQL statement and additional criteria, if desired, and enter function code `S`.

   The message `DBLOG snapshot facility started now` is displayed.

2. Execute a Natural program which contains the SQL statement specified in the **DBLOG Menu**. (Log data is written to the Natural DBLOG buffer.)

3. Display the snapshot data by entering the following command:

   ```
   TEST DBLOG Q
   ```

   Or:

   In the **DBLOG Menu**, enter function code `E`.

   A **Snapshot Report** screen for SQL statements similar to the example below appears:

   ```
   10:59:28              ***** NATURAL Test Utilities *****            2006-12-12
   User SAG                       - Snapshot Report -              Library SAG

   CU SN M Typ R SQLC/W     Library  Program  Store Clock Value   Line LV CID(Hex)
   01 01 D DB2              SAG      SAGTEST  2002/04/03 14:23:06 0150 01 01500101

   SQL Statement
   SELECT EMPNO,FIRSTNME,MIDINIT,LASTNAME,EDLEVEL,SALARY FROM DSN8510.EMP WHERE EM
   PNO > '000300' FOR UPDATE OF EDLEVEL



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

### Snapshot Report Information for SQL Statements

The following information is provided on the **Snapshot Report** screen for SQL statements:

| Column | Explanation |
|---|---|
| **CU** | Cursor number. |
| **SN** | Internal statement number. |
| **M** | Mode: `D` for dynamic or `S` for static. |
| **Typ** | Database type: DB2 or SQL/DS. |
| **R** | Only applicable if the Natural File Server for DB2 is in use.<br><br>Indicates by an asterisk in front of the corresponding statement that a reselection has been performed; if not, the column is left blank.<br><br>See also *Concept of the File Server* in the *Natural for DB2* documentation. |
| **SQLC/W** | Either the SQL return code in the SQLCODE field of the `SQLCA`, or the warning in the SQLWARN0 field of the `SQLCA` if `SQLCODE` is `0`. |
| **Library** | The library where the Natural program with the logged statement was cataloged. |
| **Program** | The name of the Natural program which contains the logged statement. |
| **Store Clock Value** | The time stamp of the Natural program which contains the logged statement. |
| **Line** | The source code line number of the logged statement. |
| **LV** | The call level of the Natural program which contains the logged statement. |
| **CID (Hex)** | The command ID of the logged statement in hexadecimal format. |

# 8 TEST DBLOG Command


- Syntax Diagrams ..... 56
- Keyword Explanations ..... 57

The Natural system command `TEST DBLOG` is used to execute DBLOG and display or delete the log records currently stored in the Natural DBLOG buffer. Note that `TEST DBLOG` does not provide any parameters to specify selection criteria. Selection criteria can only be specified in the **DBLOG Menu**.

The keywords that apply to the command `TEST DBLOG` are explained in the following syntax diagrams and tables. There are keywords that can be used to do both, activate and deactivate DBLOG (toggle effect). Activating and deactivating depends on whether or not there is data stored in the Natural DBLOG buffer as described in ***Keyword Explanations***.

For explanations of the symbols used in the syntax diagrams, refer to *System Command Syntax* in the *System Commands* documentation.

This section covers the following topics:

## Syntax Diagrams

### Activating DBLOG

```
                 [ A ] [ ON   ]
TEST DBLOG       [ D ] [ CONT ]
                 [ Q ] [ MENU ]
                       [ ?    ]
                       [ *    ]
```

### Deactivating DBLOG

```
                 [ A ] [ OFF  ]
TEST DBLOG       [ D ] [ SHOW ]
                 [ Q ] [ MENU ]
                       [ ?    ]
                       [ *    ]
```

## Keyword Explanations

| Keyword | Function |
|---|---|
| A | Default value.<br><br>Toggle function:<br><br>Activates<br>logging of Adabas commands if no data exists in the Natural DBLOG buffer.<br><br>Deactivates<br>logging of Adabas commands and displays the **DBLOG Trace** screen of the current log record if data exists in the Natural DBLOG buffer. |
| D | Toggle function:<br><br>Activates<br>logging of DL/I calls if no data exists in the Natural DBLOG buffer.<br><br>Deactivates<br>logging of DL/I calls and displays the **DBLOG Trace** screen of the current log record if data exists in the Natural DBLOG buffer. |
| Q | Toggle function:<br><br>Activates<br>logging of SQL statements if no data exists in the Natural DBLOG buffer.<br><br>Deactivates<br>logging of SQL statements and displays the **DBLOG Trace** screen of the current log record if data exists in the Natural DBLOG buffer. |
| CONT | Activates or reactivates (restarts) logging. A restart causes DBLOG to continue logging with the next program executed or run after DBLOG execution was stopped, and to add the new logs to the data that exists from previous recordings. |
| MENU | Invokes the **DBLOG Menu** which provides the options to activate or deactivate logging and to specify the commands, calls or statements to be logged; see the relevant section. |
| ? | |
| * | |
| SHOW | Deactivates logging and displays the **DBLOG Trace** screen of the current log record if data exists in the Natural DBLOG buffer. Log record data is not deleted but kept in the Natural DBLOG buffer. |
| ON | Clears the Natural DBLOG buffer and activates logging. |
| OFF | Clears the Natural DBLOG buffer and deactivates logging. |

# V INPL Utility

# 9 INPL Utility

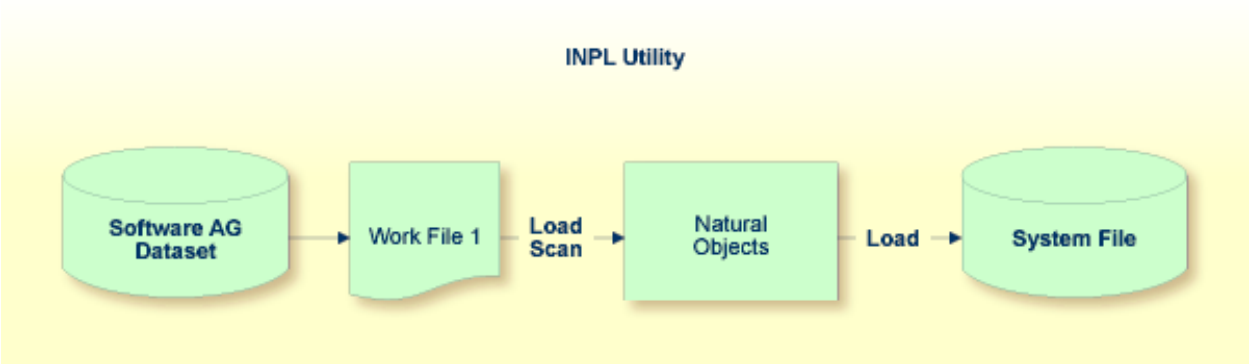

- Introducing the INPL Utility ..... 62
- Load Libraries Only ..... 68
- Load DDMs Only ..... 68
- Load Error Messages Only ..... 69
- Load All Objects ..... 69
- Replace Product Installation ..... 70
- Scan INPL File ..... 71
- Natural Security Recover ..... 71
- User Exit Routines ..... 72

The INPL utility (Initial Natural Program Load) is used to load or scan Natural objects from data sets (files) supplied by Software AG.

## Introducing the INPL Utility

The INPL utility processes Natural objects provided by Software AG.

The following diagram is a basic illustration of the INPL functionality:




The Natural objects are delivered as installation or update data sets (files) which are assigned to Work File 1. The INPL utility loads the Natural objects from Work File 1 into Natural system files.

The Natural objects include cataloged objects and source objects that are contained in libraries in the Natural system files FNAT and FUSER, or in the system file FDIC where DDMs (data definition modules) are stored.

In addition to loading Natural objects, the INPL utility provides

- a scan function to check the contents of the dataset assigned to Work File 1 and a **Natural Security Recover** function which forces initialization of the Natural Security environment.
- loading of fixes for Natural Business Services that contain Natural objects, construct models, construct frames and construct help texts if Natural Business Services version 5.3.1.8 or above is installed and the INPL function `Load All Objects` is executed.

  When fixes for Natural Business Services are loaded, PRINTER(1) is used for the Natural Business Services load report and must be defined accordingly.

If an error occurs during INPL execution, the INPL will be interrupted and terminate abnormally with Condition Code 40.

This section covers the following topics:

- Restrictions

- Special Cases
- Invoking INPL
- Options Available
- INPL Report
- Check Commands


## Restrictions

You can process only data sets (files) which are marked as “SAG system INPL data set (file)”.

## Special Cases

In all of the following cases, the INPL command must be specified using the dynamic Natural profile parameter `STACK`:

- when an INPL is to be performed on an empty FNAT system file;
- when a new upgrade INPL is to be performed;
- when an existing product installation is to be replaced;
- when an INPL is to be performed in a Natural Security environment.

## Invoking INPL

### To invoke the INPL utility

1 Enter the following Natural system command:

```
INPL
```

An INPL menu similar to the example below is displayed:

```
18:32:19                ***** NATURAL INPL UTILITY *****            2005-04-21
User: SAG                                                   Library: SYSTEM  ↩

                     Code   Function

                       L    Load Libraries Only
                       D    Load DDMs Only
                       E    Load Error Messages Only
                       B    Load All Objects
                       P    Replace Product Installation
                       S    Scan INPL File
                       R    Natural Security Recover
                       ?    Help
                       .    Exit

         Code ........ B
         Replace ..... Y  (Y/N/P/O)   Load Except . N  (Y/N)
         DDM Name ....
         Library .....
         Object Name .                Date ........             (YYYY-MM-DD)
         Check Date .. N  (Y/N)       Number ...... 0
```

2 From the INPL menu, you can choose one of the following functions by entering the corresponding function code in the **Code** field:

- **Load Libraries Only**
- **Load DDMs Only**
- **Load Error Messages Only**
- **Load All Objects**
- **Replace Product Installation**
- **Scan INPL File**
- **Natural Security Recover**

For detailed information on these functions, refer to the corresponding sections.

3 Modify or complete the remaining input fields as described in *Options Available*.

4 Choose ENTER to confirm your entries.

## Options Available

The following section describes the input fields on the INPL menu where you can specify one or more parameters as object selection criteria for the INPL function specified in the **Code** field. The use of a parameter depends on the respective function as indicated in the relevant documentation sections.

| Field | Description |
|---|---|
| **Replace** | Specifies whether the Natural objects to be processed are to replace any that already exist on the system files.<br><br>Possible settings are:<br><br>`Y` All existing objects are replaced. This is the default setting.<br>`N` Existing objects are *not* replaced.<br>`P` All existing objects are replaced. Additionally, existing Natural or Natural add-on product installations are replaced depending on the corresponding **check commands** (see below) executed during INPL processing.<br>`O` Resets the owner information of specified objects. Only applies to the function **Natural Security Recover**.<br><br>See also **Check Date** to replace only objects that are older than the objects to be processed.<br><br>If you use the function **Natural Security Recover**, you can enter option `O` in this field to reset the owner information of specified objects. |
| **DDM Name** | The name of a DDM or a range of names.<br><br>If you enter a value that ends with an asterisk (*), each DDM with a name that starts with the specified value is processed. If only an asterisk (*) is entered or if this field is empty, all DDMs are processed. |
| **Library** | The name of a library or a range of names.<br><br>If you enter a value that ends with an asterisk (*), each library with a name that starts with the specified value is processed. The library name is mandatory if **Object Name** is specified. |
| **Object Name** | The name of a Natural object (except DDMs) or a range of names.<br><br>If the value ends with an asterisk (*), each object with a name that starts with the specified value is processed.<br><br>If this field is empty, all objects contained in the library specified in the **Library** field are processed. |
| **Check Date** | Specifies whether existing Natural objects are to be replaced depending on their time stamp.<br><br>This parameter has no effect if **Replace** is set to `N`.<br><br>Possible settings are: |

| Field | Description |
|---|---|
| | Y Only objects which are older than the objects of the same name are replaced. An object is older if it was saved or cataloged before the object to be loaded.<br>N All objects are replaced. This is the default setting. |
| **Load Except** | Specifies whether to exclude Natural objects from processing.<br><br>This parameter does not apply to error messages.<br><br>Possible settings are:<br><br>Y All objects are processed except for the objects specified in the fields **DDM Name**, **Library** and/or **Object Name**.<br>N No exceptions; all objects are processed. This is the default setting.<br><br>**Examples of load exceptions:**<br><br>All libraries except the library ABC are loaded:<br>**Code** = L<br>**Library** = ABC<br><br>All DDMs with a prefix other than XY are loaded:<br>**Code** = D<br>**DDM Name** = XY*<br><br>All objects contained in libraries with a prefix other than AB and all DDMs with a prefix other than CD are loaded:<br>**Code** = B<br>**Library** = AB*<br>**DDM Name** = CD* |
| **Date** | Restricts processing to Natural objects which were saved or cataloged on or after the date entered in this field.<br><br>The date must be entered in the format *YYYY-MM-DD* (*YYYY* = year, *MM* = month, *DD* = day). |
| **Number** | Limits processing of Natural objects to a specified number. All objects are counted which are loaded or scanned according to the selection criteria specified in the INPL menu.<br><br>If the number of objects processed has reached the value entered in the **Number** field, processing is terminated with a corresponding message. |

### INPL Report

When the selected INPL function is complete, a corresponding INPL report is displayed on the screen.

### Check Commands

The INPL utility processes internal check commands which are executed automatically when performing an INPL. Check commands are used to control the load or the scan process and react on certain events. The check commands which are executed during an INPL are written into the INPL report.

The parameters used by the check command to react on an event are, for example, `STOP`, `LOAD`, `CONTINUE`, `SKIP`, `USERLOAD` or `USERCONTINUE`. You can display all parameters by using the **Scan INPL File** function. These parameters, for example, are used to verify that the version of a product to be installed is higher than the version of the product already installed.

You cannot modify a check command but you can influence the effect of the parameters `USERLOAD` and `USERCONTINUE` by selecting the function Replace Product Installation (function code `P`) or by setting the **Replace** option to `P`.

`USERLOAD` (or `USERCONTINUE`) means: Load (or continue loading) only if function code `P` has been selected or if the **Replace** option is set to `P`.

**Example of USERLOAD and USERCONTINUE**

In the example of a check command below, the Natural version currently installed is checked before the INPL is performed:

```
CHECK VERSION NAT vrs LT USERLOAD EQ LOAD GT USERLOAD
```

If the Natural (`NAT`) version installed is below (`LT`) or above (`GT`) 421, the INPL function is only performed if function code `P` has been specified or if the **Replace** option has been set to `P`. Otherwise, the INPL utility stops or terminates with Condition Code 40 in batch and writes a corresponding message into the INPL report.

If the version is equal to 421, the INPL function is always performed, irrespective of whether function code `P` has been specified or the **Replace** option has been set to `P`.

## Load Libraries Only

This function of the INPL utility is used to load Natural cataloged objects and source objects into specified libraries in the Natural system file FNAT or FUSER.

### To load libraries

1 In the INPL menu, enter function code `L`. You can specify parameters to be valid during execution of this function:

- **Replace** (Y/N)
- **Load Except** (Y/N)
- **Library**
- **Object Name**
- **Date** (YYYY-MM-DD)
- **Check Date** (Y/N)
- **Number**

For detailed information on these parameters, refer to ***Options Available*** in the section *Introducing the INPL Utility*.

2 Confirm your entries.

When the function is complete, a corresponding **INPL report** (see the section *Introducing the INPL Utility*) is output.

## Load DDMs Only

This function of the INPL utility is used to load DDMs into the system file FDIC.

### To load DDMs

1 In the INPL menu, enter function code `D`. You can specify parameters to be valid during execution of this function:

- **Replace** (Y/N)
- **Load Except** (Y/N)
- **DDM Name**
- **Number**

For detailed information on these parameters, refer to ***Options Available*** in the section *Introducing the INPL Utility*.

2 Confirm your entries.

When the function is complete, a corresponding **INPL report** (see the section *Introducing the INPL Utility*) is output.

## Load Error Messages Only

This function of the INPL utility is used to load user-defined error messages or system error messages into specified libraries in the Natural system file FUSER or FNAT respectively.

### To load error messages

1 In the INPL menu, enter function code `E`. You can specify parameters to be valid during execution of this function:

- **Replace** (Y/N)
- **Library**

For detailed information on these parameters, refer to ***Options Available*** in the section *Introducing the INPL Utility*.

2 Confirm your entries.

When the function is complete, a corresponding **INPL report** (see the section *Introducing the INPL Utility*) is output.

## Load All Objects

This function of the INPL utility is used to load all Natural objects (including error messages and DDMs) into the libraries indicated in Work File 1. DDMs are loaded into the system file FDIC.

### To load all objects

1 In the INPL menu, enter function code `B`. You can specify parameters to be valid during execution of this function:

- **Replace** (Y/N)
- **Load Except** (Y/N)
- **DDM Name**

- **Library**
- **Object Name**
- **Date** (YYYY-MM-DD)
- **Check Date** (Y/N)
- **Number**

For detailed information on these parameters, refer to ***Options Available*** in the section *Introducing the INPL Utility*.

2 Confirm your entries.

When the function is complete, a corresponding **INPL report** (see the section *Introducing the INPL Utility*) is output.

## Replace Product Installation

In addition to the function **Load All Object**, this function replaces any existing Natural or Natural add-on products installed in the current system environment. The replacement of existing product installations depends on the execution of the corresponding **check commands** described in the section *Introducing the INPL Utility*.

### To load all objects and replace existing product installations

1 In the INPL menu, enter function code `P`. You can specify parameters to be valid during execution of this function:

- **Replace** (Y/N)
- **Load Except** (Y/N)
- **DDM Name**
- **Library**
- **Object Name**
- **Date** (YYYY-MM-DD)
- **Check Date** (Y/N)
- **Number**

For detailed information on these parameters, refer to ***Options Available*** in the section *Introducing the INPL Utility*.

2 Confirm your entries.

When the function is complete, a corresponding **INPL report** (see the section *Introducing the INPL Utility*) is output.

## Scan INPL File

This function of the INPL utility is used to scan the contents of the data set (file) assigned to Work File 1.

### To scan an INPL File

1 In the INPL menu, enter function code `S`. You can specify parameters to be valid during execution of this function:

- **Load Except** (Y/N)
- **DDM Name**
- **Library**
- **Object Name**
- **Date** (YYYY-MM-DD)
- **Number**

For detailed information on these parameters, refer to ***Options Available*** in the section *Introducing the INPL Utility*.

2 Confirm your entries.

When the function is complete, a corresponding **INPL report** (see the section *Introducing the INPL Utility*) is output.

## Natural Security Recover

This function of the INPL utility is used to force initialization of the Natural Security environment.

The following options are provided:

- Reset Environment

- Remove Owners

### Reset Environment

**Caution:** Execution of this function will reset the user profile DBA and the library profile SYSSEC as well as the link between these two objects as they were after the initial installation; all other links to the library SYSSEC will be canceled. Other Natural Security profiles and links will not be modified. Contact Software AG technical support for further information.

**To reset the environment**

- In the INPL menu, enter function code `R`.

### Remove Owners

**To remove owners**

- In the INPL menu, enter function code `R` and enter an `0` in the **Replace** field to reset the owner information of specified objects.

## User Exit Routines

An INPL user exit routine is supplied as source object INPLSX*nn* in the Natural system library SYSLIB where *nn* denotes the ID of the user exit routine.

**To activate a user exit routine**

1. Copy the source code from SYSLIB into a user library.
2. Catalog it under the name `INPLUX`*nn*.
3. Copy it back into the Natural system library SYSLIB.

**Note:** The source object that you might have modified, and the cataloged object of the user exit routine are renamed to avoid them to be overwritten by an update installation.

The following user exit routines are available:

| Name | Function |
| --- | --- |
| INPLUX01 | Prevent error message texts to be replaced. |

### INPLUX01

You can use this user exit to define ranges for error messages (user defined or Natural system error messages) that cannot be replaced during an INPL session. For further details, see the source of INPLSX01 in the Natural system library SYSLIB.

# VI NATPAGE Utility - Screen Capturing

# 10 NATPAGE Utility - Screen Capturing

The utility NATPAGE is used to capture screen output data (maps and reports) during a Natural session. The term screen in this context means the contents of the page buffer; that is, the logical page output by Natural.

The screen captures are stored in the Natural scratch-pad file as described in the relevant section in the *Operations* documentation.

The maximum number of screens that can be captured is determined by the session parameter `PD`, which is described in the *Parameter Reference* documentation.

The NATPAGE utility consists of the following Natural terminal commands:

| Command | Function |
|---|---|
| `%P` | Activates NATPAGE and captures the contents of the current screen and all subsequent screens. Screens captured previously are deleted. |
| `%I` | Activates NATPAGE and captures the contents of the current screen. |
| `%O` | Deactivates NATPAGE. |
| `%S` | Resumes NATPAGE. |
| `%E` | Displays the screens captured with NATPAGE. |

See the *Terminal Commands* documentation for a detailed description of these terminal commands.

# VII NATRJE Utility - Natural Remote Job Entry

# 11 NATRJE Utility - Natural Remote Job Entry


- General Information on NATRJE ..... 82
- Calling NATRJE from a Natural Program ..... 83
- NATRJE Return Codes ..... 88
- NATRJE Features Applicable to openUTM/TIAM ..... 89

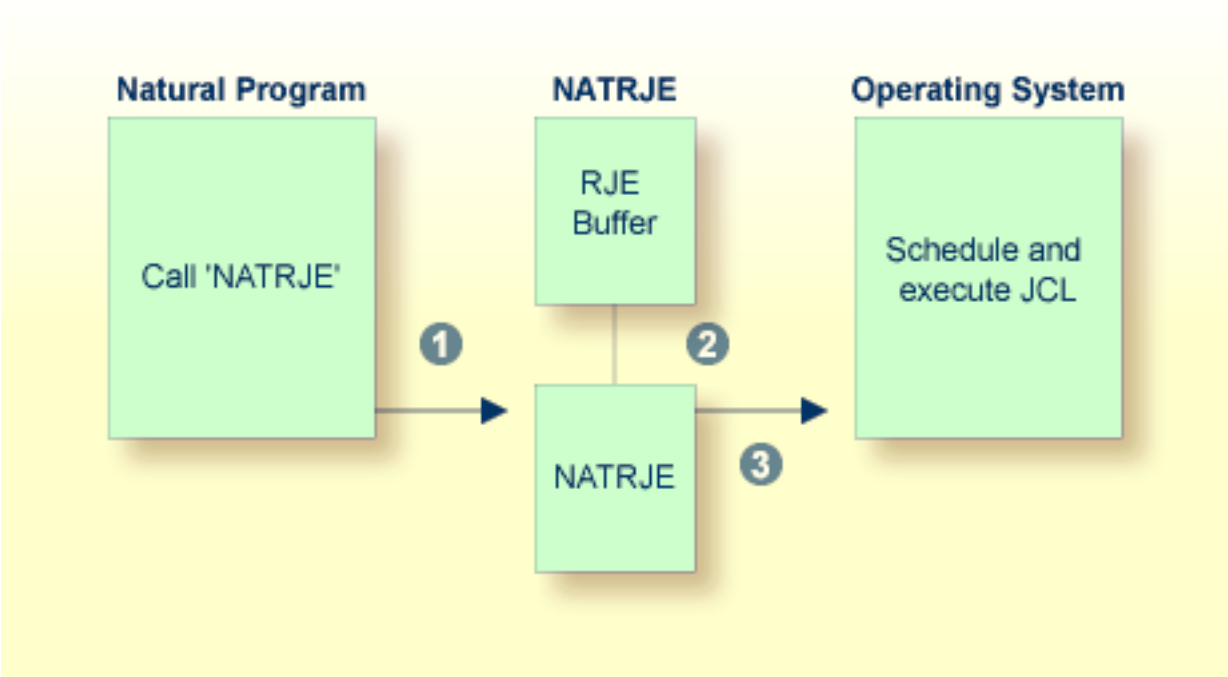
The NATRJE utility (Natural Remote Job Entry) can be used to submit JCL cards from a Natural program to the operating system for scheduling and execution. For example, it is possible to start a Natural batch job with NATRJE.

**Related Topic:**

- *NREXPG - User Exit for NATRJE* in the *Operations* documentation

## General Information on NATRJE

The following graphic is a simplified functional diagram of the NATRJE utility:




### Legend

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

3. NATRJE transfers the JCL cards to the operating system internal job queue for scheduling and execution by the operating system.

   Note for BS2000:

   When the job generation is complete, NATRJE transfers the JCL cards to a BS2000 file, which is generated by NATRJE. This is a SAM file that contains the generated JCL cards and is submitted by using the `ENTER`, `ENTER-JOB` or `ENTER-PROCEDURE` command.

## Calling NATRJE from a Natural Program

This section covers the following topics:

- Invoking NATRJE
- Example Programs

### Invoking NATRJE

**To invoke the NATRJE utility**

- In the Natural program that invokes the NATRJE utility, specify a `CALL` statement by using the following syntax:

```
CALL 'NATRJE' parm1 parm2 parm3 parm4 [parm5]
```

**Note:** *parm5* only applies to BS2000.

The parameters specified in the `CALL` statement are explained in the following table:

| Parameter | Explanation | |
|---|---|---|
| *parm1* | The starting JCL card of the table which contains one or more 80-character JCL cards to be submitted. | |
| *parm2* | A 4-byte binary field which contains the number of 80-character JCL cards to be submitted. | |
| *parm3* | A 1-byte alphanumeric field used to indicate if all JCL cards have been submitted: | |
| | | |
| | ' ' | Not the last call for the current job. A further JCL card follows with the next `CALL` statement. The JCL cards are collected into the RJE buffer. |
| | | |

| Parameter | Explanation | |
|---|---|---|
| | B | BS2000 and z/OS only: Last call for the current job.<br><br>Under BS2000: The job is generated, written to the file, but not started automatically.<br><br>Under z/OS (batch and TSO, IMS TM and CICS): The job is written to the internal reader data set but not submitted. If function L (see below) is called subsequently, the internal reader is closed and the job(s) are submitted. In addition, the internal reader is closed and the job is submitted either on a screen I/O (IMS TM) or during session termination (z/OS batch, TSO and IMS TM). |
| | | |
| | C | Flush the current job. The job is not submitted to the system. Under BS2000, no file is created. |
| | | |
| | L | Last call for the current job. The job is submitted to the system. |
| | | |
| | BS2000 environments: see ***Additional Values for parm3***. | |
| *parm4* | A 2-byte binary field in which NATRJE returns a response code. | |
| *parm5* | Optional parameter for BS2000 only.<br><br>A 1880-byte (maximum) alphanumeric field used to define a list of parameters as highlighted in ***Example Program 3 for BS2000***.<br><br>You can replace the default file names generated by Natural by user-defined names if *parm5* starts with a FROM-FILE parameter definition.<br><br>**Note:**<br><br>1. Natural only checks the syntax of the FROM-FILE parameter definition in *parm5*. It is the responsibility of the user to validate the syntax of all other definitions contained in *parm5*.<br>2. *parm5* is not evaluated for the ISP ENTER command. | |

## Example Programs

This section provides Natural example programs that submit JCL cards:

- Example Program for z/OS
- Example Program for z/VSE
- Example Program 1 for BS2000
- Example Program 2 for BS2000

- Example Program 3 for BS2000

### Example Program for z/OS

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

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

### Example Program for z/VSE

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

```
DEFINE DATA LOCAL
   01 COUNT  (B4)
   01 FLAG   (A1)
   01 RETHEX (B2)
   01 CARDS  (A240)
   01 REDEFINE CARDS
      02 CARD1 (A80)
      02 CARD2 (A80)
      02 CARD3 (A80)
  END-DEFINE
  MOVE '* $$ JOB JNM=DSERV,CLASS=O,DISP=D' TO CARD1
  MOVE '* $$ LST CLASS=A,DISP=D' TO CARD2
  MOVE '// JOB DSERV TO DSERV SOURCE MEMBERS' TO CARD3
  MOVE 3 TO COUNT
  CALL 'NATRJE' CARDS COUNT FLAG RETHEX
```

```
  PERFORM RETCODE-CHECK
  MOVE '// EXEC PROC=NATSPLP' TO CARD1
  MOVE '// EXEC  DSERV' TO CARD2
  MOVE ' DSPLYS SD' TO CARD3
  MOVE 3 TO COUNT
  CALL 'NATRJE' CARDS COUNT FLAG RETHEX
  PERFORM RETCODE-CHECK
  MOVE '/*' TO CARD1
  MOVE '/&' TO CARD2
  MOVE '* $$ EOJ' TO CARD3
  MOVE 3 TO COUNT
  MOVE 'L' TO FLAG
  CALL 'NATRJE' CARDS COUNT FLAG RETHEX
  DEFINE SUBROUTINE RETCODE-CHECK
  IF RETHEX NE H'0000'
    WRITE 'ERROR from NATRJE:' RETHEX
  STOP
  END-IF
  END-SUBROUTINE
  END
```

#### Example Program 1 for BS2000

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

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

```
   ...
   END-IF
  MOVE 'RUNPGM' TO CARD1
  MOVE 'FIN' TO CARD2
  MOVE '/LOGOFF' TO CARD3
  MOVE 3 TO COUNT
  MOVE 'L' TO FLAG
  CALL 'NATRJE' CARDS COUNT FLAG RETHEX
   ...
   ...
   ...
  END
```

#### Example Program 2 for BS2000

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

```
DEFINE DATA LOCAL
   01 COUNT  (B4)
   01 FLAG   (A1)
   01 RETHEX (B2)
   01 CARD1 (A80)
   01 CARD2 (A80)
   01 CARD3 (A80)
   01 CARD4 (A80)
  ...
   01 CARD9 (A80)
  END-DEFINE
  MOVE '/LOGON' TO CARD1
  MOVE '/SYSFILE SYSDTA=(SYSCMD)' TO CARD2
  ...
  MOVE '/LOGOFF' TO CARD9
  MOVE 9 TO COUNT
  MOVE 'L' TO FLAG
  CALL 'NATRJE' CARD1 COUNT FLAG RETHEX
  ...
  END
```

#### Example Program 3 for BS2000

The following is a Natural example program that submits, in one call to NATRJE, a JCL procedure with two `ENTER-PROCEDURE` command parameters.

```
DEFINE DATA LOCAL
 01 COUNT  (B4)
 01 FLAG   (A1)
 01 RETHEX (B2)
 01 CARDS (A80/1.8)
 01 01 SDF-COM (A80) INIT <'JOB-NAME=RJEPTEST,RESOURCES=*PAR(RUN-PRIO=230)'>
END-DEFINE
CARDS (1) := '/BEGIN-PROCEDURE LOGGING=COMMANDS'
CARDS (2) := '/ASSIGN-SYSOUT  TO=RJETEST.OUT'
CARDS (3) := '/SHOW-JOB-STATUS'
CARDS (4) := '/SHOW-FILE-ATTRIBUTES FILE-NAME=*ALL,SELECT=BY-ATTRIBUTE'
CARDS (5) := '/ASSIGN-SYSOUT  TO=*PRIMARY'
CARDS (6) := '/EXIT-PROCEDURE'
COUNT := 6                                    /* NUMBER OF CARDS */
FLAG := 'P'                                   /* LAST CALL OF NATRJE */
CALL 'NATRJE' CARDS(*) COUNT FLAG RETHEX SDF-COM
WRITE '=' RETHEX
END
```

## NATRJE Return Codes

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

- Return Codes Common to all Environments
- Additional Return Codes for CICS and Batch under z/VSE
- Additional Return Codes for CICS under z/OS
- Additional Return Codes under BS2000

### Return Codes Common to all Environments

| Return Code/Hexadecimal | Return Code/Decimal | Explanation |
|---|---|---|
| 00 | 00 | Normal return. |
| 04 | 04 | NATRJE utility not available. |
| 08 | 08 | NATRJE utility disabled; a possible reason is that the `RJESIZE` profile parameter is set to `0` (see also the *Parameter Reference* documentation). |
| 0C | 12 | Invalid number of JCL cards. |
| 10 | 16 | Invalid function code. |
| 14 | 20 | No RJE buffer space available. |
| 18 | 24 | Invalid number of parameters. |
| 1C | 28 | I/O error during submit. |

| Return Code/Hexadecimal | Return Code/Decimal | Explanation |
|---|---|---|
| 20 | 32 | Job flushed by user exit NREXPG (see *NREXPG - User Exit for NATRJE* in the *Operations* documentation). |

### Additional Return Codes for CICS and Batch under z/VSE

| Return Code | Explanation |
|---|---|
| `ffrr` | `ff` is the XPCC request function code and `rr` the associated return code. |

### Additional Return Codes for CICS under z/OS

| Return Code | Explanation |
|---|---|
| 01*nn* | CICS WRITEQ TD failure; *nn* is the CICS response code in hexadecimal format. |
| 01*nn* | CICS CLOSE TD failure; *nn* is the CICS response code in hexadecimal format. |

### Additional Return Codes under BS2000

| Return Code | Explanation |
|---|---|
| 9001 | No RJE buffer found. |
| 9002 | No buffer space available. |
| 9003 | Missing `LOGON` command. |
| 9004 | Only `LOGON` cards generated. |
| 9005 | Too many `LOGON` parameters. |
| 9006 | File name in `FROM-FILE` parameter invalid or longer than 54 characters. |
| D010 | Error in ENTER macro. |
| D*xxx* | Operating system error:<br><br>The error message is sent directly to the user program; the BS2000 `HELP` command provides additional information. |

## NATRJE Features Applicable to openUTM/TIAM

This section covers the following topics:

- SDF Command SET-LOGON-PARAMETERS
- Additional Values for parm3
- Name of BS2000 File

- Optional CALL Parameter for SDF ENTER Commands

### SDF Command SET-LOGON-PARAMETERS

You can replace the ISP command `LOGON` by the SDF command `SET-LOGON-PARAMETERS` in the first JCL card of the job to be executed. However, the following restrictions apply when using the SDF command:

- The only supported command abbreviation for the `SET-LOGON-PARAMETERS` command is `STLGP`;
- If the abbreviation `STLGP` is used, the value of *`parm3`* must be set to `S`;
- The values `A` and `T` for *`parm3`* are not supported.

**Examples of ISP and SDF Commands:**

ISP command

```
/.job-id LOGON user-id,account-number,'password'
```

and corresponding SDF command:

```
/.job-id STGLP user-id,account-number,'password'
```

SDF command with additional keyword operands:

```
/.job-id STGLP user-id,account-number,'password', -
/RESOURCES=*PARAMETERS(RUN-PRIORITY=220)
```

### Additional Values for parm3

| Value | Explanation |
|---|---|
| `A` | Combination of values `T` and `E` (see below). |
| `E` | The job is generated and completed. Before submission to the BS2000 operating system, the parameter `ERASE=YES` is added to the `ENTER` parameter. |
| `T` | The job is generated and completed. Before submission to the BS2000 operating system, a time limit is calculated using the Natural `MT` parameter (see also the *Parameter Reference* documentation). If `MT` is set to `0`, the time limit is generated as `NTL`. The calculated time limit is added to the `ENTER` parameter via the `TIME=`*`operand`*. |
| `S` | This value must be set if the abbreviation `STLGP` for the SDF command `SET-LOGON-PARAMETERS` is used. |
| `J` | The SDF command ENTER-JOB is used for job submission. |
| `P` | The SDF command ENTER-PROCEDURE is used for job submission. |

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

### Name of BS2000 File

The name of the BS2000 file created by NATRJE for the JCL cards is as follows:

```
E.DDMMYY.HHMMSSSS.program-name.user-id - if the ISP command ENTER is used for job ↩
submission
J.DDMMYY.HHMMSSSS.program-name.user-id - if the SDF command ENTER-JOB is used for ↩
job submission
P.DDMMYY.HHMMSSSS.program-name.user-id - if the SDF command ENTER-PROCEDURE is used ↩
for job submission
```

The parameters are explained in the following table.

| Parameter | Explanation |
|---|---|
| DD | The day of the file creation. |
| MM | The month of the file creation. |
| YY | The year of the file creation. |
| HH | The hour of the file creation. |
| MM | The minute of the file creation. |
| SSSS | The seconds and milliseconds of the file creation. |
| *program-name* | The name of the Natural program that creates the file. |
| *user-id* | The corresponding Natural user ID. |

### Optional CALL Parameter for SDF ENTER Commands

Under BS2000, the CALL statement provides the option to submit a JCL procedure along with an ENTER-JOB or ENTER-PROCEDURE command. For details, see the *parm5* parameter described in *Invoking NATRJE* and ***Additional Values for parm3***.

For an example of a program call, see ***Example Program 3 for BS2000***.

# VIII Object Handler

The Object Handler is designed to process Natural and non-Natural objects for distribution in Natural environments. This is done by unloading the objects in the source environment into work files and loading them from work files into the target environment.

The *Object Handler* documentation is organized in the following parts:

| | |
|---|---|
| **General Information on the Object Handler** | Invoking the Object Handler in batch or online mode; applying Natural Security. |
| **Functions** | Using the Object Handler menu functions: unload, load, restart load, scan, view, find and administration. |
| **Object Specification** | Specifying the objects to be processed with Object Handler menu functions: **Natural Library Objects**, **Natural System Error Messages**, **Natural Command Processors**, **DDMs**, **Natural-Related Objects**, **FDTs**. |
| **Settings** | Specifying option and parameter settings for Object Handler menu functions. |
| **Workplans** | Using standard procedures to execute Object Handler functions. |
| **Name, Date and Time Specification** | Specifying names, dates, times and ranges. |
| **Work Files** | Work files used by the Object Handler. |
| **Direct Commands** | Using direct commands to perform Object Handler functions. |
| **Batch Condition Codes and User Exit Routines** | Condition codes and user exit routines provided in batch mode. |
| **Tools** | Displaying status information and setting trace and report options. |
| **Profile Settings** | Setting up a profile to define individual defaults and standard procedures. |
| **Migration from NATUNLD/NATLOAD and SYSTRANS to the Object Handler** | Migrating from the utilities NATUNLD/NATLOAD and SYSTRANS to the Object Handler. |

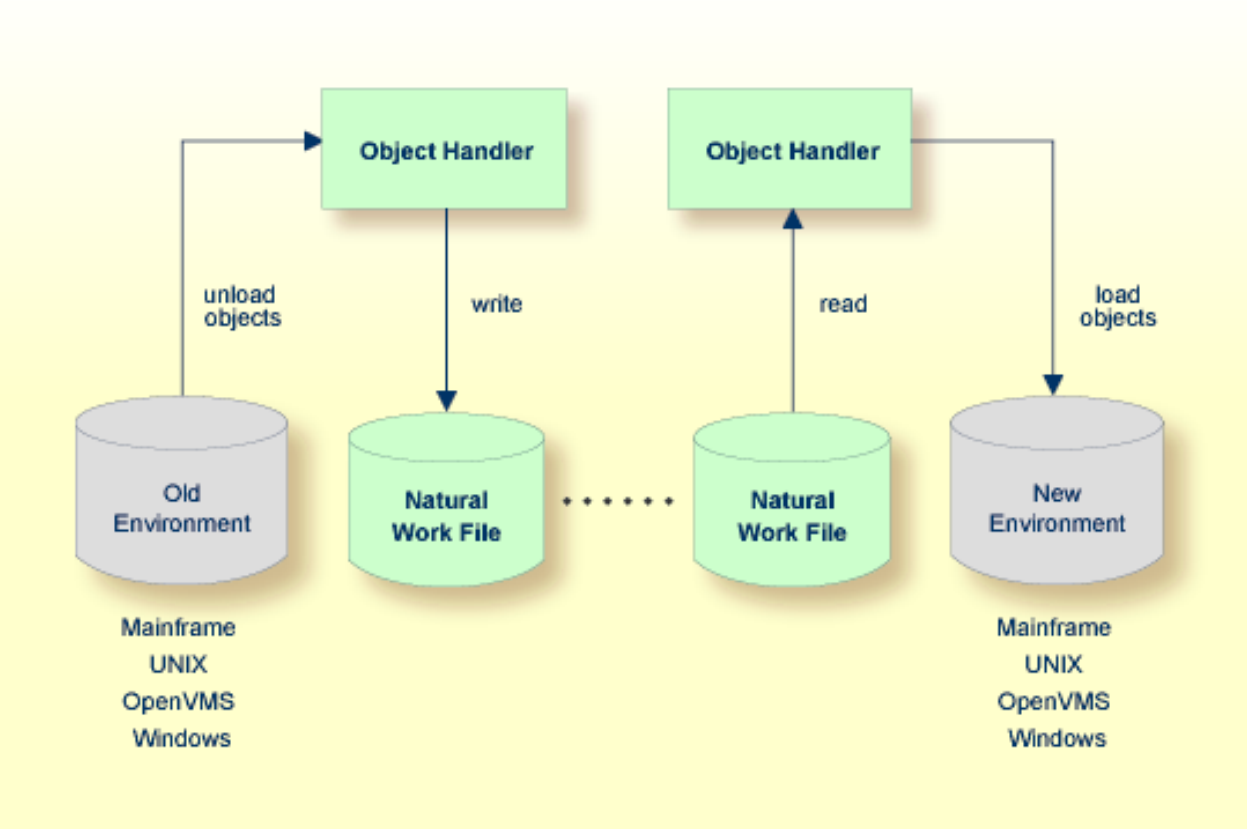
# 12 General Information on the Object Handler


- Principles of Object Transfer ..... 96
- Invoking the Object Handler ..... 97
- Batch or Direct Command Calls ..... 99
- Issuing Object Handler Commands from a Natural Program ..... 100
- Text Members for Reports, Restarts and Traces ..... 100
- Natural Security ..... 101
- Standard PF Keys ..... 101

The Object Handler consists of the utility SYSOBJH which is located in the Natural system library SYSOBJH, and the direct command interface. Additionally, the Application Programming Interface OBJHAPI is provided for executing Object Handler functions from a Natural program.

## Principles of Object Transfer

The diagram below illustrates how the Object Handler transfers objects by unloading them from the source environment into work files and loading them from work files into the target environment. If required, an application protocol such as FTP can be used for transferring work files from source to target environments.




This section covers the following topics:

- Transfer Environment and File Security
- Objects Processed by the Object Handler

- Formatting Options

### Transfer Environment and File Security

An old or a new environment is an FNAT, FUSER or FDIC system file contained in an Adabas database or a VSAM file system on a mainframe, or in the file system on a UNIX, an OpenVMS or a Windows platform. Natural objects on the FNAT or FUSER system file can be contained in libraries as indicated in the following section.

The file security (that is, passwords and cipher codes) relates to the security that has been defined for a system file in an Adabas or a VSAM environment. If file security has been defined for a system file, you need to specify a password, cipher code and/or VSAM name for the source and/or target system file required before you perform an Object Handler function. Otherwise, Adabas or VSAM will issue an appropriate error message. You do not have to provide security information for the default system files assigned to the Natural session at the start of the Object Handler.

### Objects Processed by the Object Handler

The Object Handler transfers Natural source objects (also referred to as saved objects) and cataloged objects which are contained in Natural libraries, Natural error messages, Natural command processor sources, Natural DDMs (data definition modules), Natural-related objects, and Adabas FDTs (Field Definition Tables).

### Formatting Options

You can transfer data of binary or text format, depending on the source and target environment where the objects are processed.

Binary format can be used for source objects and cataloged objects (including DDMs), error messages, Natural command processor sources, Natural-related objects and Adabas FDTs.

Text format applies to source objects, Natural command processor sources, error messages, DDMs and Adabas FDTs. You can only transfer text data between mainframe and UNIX/OpenVMS/Windows platforms. You can transfer binary data between identical platforms.

## Invoking the Object Handler

To invoke the Object Handler you can either use menu functions or direct commands.

**To invoke the Object Handler online from any Natural library**

1 Enter the following system command (online from any Natural library):

```
SYSOBJH
```

Depending on the input mode, either the Object Handler **Main Menu** or the Object Handler **Compact Menu** is displayed. See ***Compact Mode***. You can change the input mode either by the `SET INPUT` command (described in *Direct Commands*) or by changing the `INPUT-MODE` parameter (described in *Profile Settings*). The **Main Menu** of the Object Handler provides the following options:

- Unload
- Load
- Scan
- View
- Administration

See the section ***Functions*** for descriptions of these functions, and how they can be processed (in advanced-user mode or by using wizards).

2 Select a function by choosing one of the following methods:

Enter any character in the input field next to the item that corresponds to the function required.

Or:

Choose the PF key that corresponds to the function required.

Or:

In the Command line, enter the Object Handler command that corresponds to the function required. For information on the commands provided, see the section ***Direct Commands***.

### To invoke the Object Handler in batch or direct command online mode

- Enter the system command `SYSOBJH` followed by a direct command as described in ***Batch or Direct Command Calls*** and in ***Direct Commands***.

  After execution of a direct command, you can enter either another direct command or a period (.) to exit the Object Handler.

## Batch or Direct Command Calls

Several commands can be issued to the Object Handler online or in batch mode. The last command in the command sequence must be a period (`.`), `STOP`, `END`, `QUIT` or `FIN`, where `FIN` ends the Natural session.

The section covers the following topics:

- Batch Mode
- Online Mode

### Batch Mode

The commands to the Object Handler are read from standard input. Each command can be separated into a maximum of 20 command parts/strings by entering input delimiters (session parameter `ID`) after any keyword or keyword value. Each command part/string must not exceed 248 bytes.

If the command is longer than a single line, at the end of every line except the last that belongs to the command, enter the character defined with the session parameter `CF` (default is `%`) This indicates continuation on the next line. However, this is only possible if you specify the command `SYSOBJH` in a line by itself. That is, you cannot use `CF`, if you enter `SYSOBJH` in the same line where a multi-line command starts. In addition, we recommend that you set the `LS` profile parameter to `250`.

**Example (assuming ID is set to ,):**

```
SYSOBJH
UNLOAD PROG* LIBRARY MYLIB1, OBJTYPE N,%
WITH NEWLIBRARY MYLIB2%
WHERE REPORT TRANSFER
STOP
```

**Related Topics:**

- *Direct Commands*
- *Natural in Batch Mode - Operations* documentation

### Online Mode

The command to the Object Handler in the Command line can consist of up to 20 command parts.

**Example:**

```
SYSOBJH UNLOAD * LIB EXAMPLE WHERE TRANSFER
```

## Issuing Object Handler Commands from a Natural Program

You can issue commands to the Object Handler with a Natural program by using the OBJHAPI Application Programming Interface, which is supplied as a subprogram in the Natural system library SYSOBJH. For the parameters required and examples, see the Natural program DOC-API supplied in the library SYSOBJH.

## Text Members for Reports, Restarts and Traces

Report, restart and trace data created by the Object Handler are stored as Natural text members (Natural objects of the type Text) in the Workplan library. The Object Handler generates names for text members that have not been explicitly specified in the **Options** window. The names generated are a combination of the weekday and the time. For example: a member with the name 21415568 was created on Tuesday (the second day of the week) at 14:15:56,8.

You can specify the Workplan library in which the text members are stored by using the `Workplan-Library` profile option of the Object Handler described in ***Profile Settings***. If your Natural session uses a read-only FNAT or FUSER system file as specified with the `ROSY=ON` profile parameter (see the *Parameter Reference* documentation), the text members are stored in a scratchpad file (see the *Operations* documentation) under one of the following conditions:

- The database ID and file number specified for the Workplan library is identical to the current FNAT or FUSER system file.
- No database ID and file number are specified for the Workplan library.

## Natural Security

The use of the Object Handler under Natural Security requires that utility profiles be defined for it in Natural Security. At least, a default profile must be defined. For information on utility profiles, see the section *Protecting Utilities* in the *Natural Security* documentation.

If Natural Security is installed, the Object Handler checks the SYSOBJH utility profiles in Natural Security to find out whether the requested function and the parameter settings are allowed.

Should a Natural Security error occur during the load function, the following applies:

- If the **Write report** option is set, in online mode, the error message is written to the report file and processing continues for the current load command.
- If the **Write report** option is set, in batch mode, the error message is written to the report file and the Object Handler terminates after the load command where the error occurred has finished processing.
- If the **Write report** option is not set, an error message is issued and the load command is terminated.

## Standard PF Keys

The following PF keys are available on all full-screen maps:

| PF Key | Explanation |
|---|---|
| PF1 | Invokes the help function for the field at which the cursor is positioned. |
| PF3 | Exits the current screen and returns to the previous screen. |
| PF6 | Goes to the top of a list. |
| PF7 | Scrolls up one page in a list.<br><br>On wizard screens: goes back one screen/step. |
| PF8 | Scrolls down one page in a list.<br><br>On wizard screens: goes to the next screen/step. |
| PF9 | Goes to the bottom of a list. |
| PF10 | Invokes the **Commands** menu to select commands for navigation purpose and to assign special settings. See also ***Commands for Navigation and Special Functions*** in *Direct Commands*. |
| PF12 | Cancels the current function. |
| PF20 | Lists all active programs of the Object Handler. This can be helpful information for reporting technical problems to Software AG. |

# 13 Functions

This section describes the main functions provided by the Object Handler.

You can take advantage of the Object Handler wizards to guide you through the steps required to execute the unload, load and scan functions. The wizards are activated by default unless you change default settings. If you prefer the unload, load or scan mode for the experienced user instead, select the field next to **Advanced user** in the **Main Menu** or use the internal command `SET INPUT-MODE A`. Additionally, if you are an experienced user, you can use compact input mode for the unload, load or scan functions by entering the internal command `SET INPUT-MODE C`. You can also set the default input mode (wizard, advanced or compact) by using the appropriate object Handler profile option **Input-Mode**. See also the section *Profile Settings*.

**Tip:** You can create standard procedures to define recurring settings and object specifications which automate the processing of the unload, load or scan function, see *Workplans*.

This section covers the following topics:


**Wizards**

**Advanced User**

**Compact Mode**

**Restart Load**

**View**

**Find**

**Administration**

**Change the Workplan Library**

**Select System File**

**Select Library**

**List and Select Workplan**

**Select System Error Messages**

### Select Objects

**Notes:**

1. The topic *Change the Workplan Library* is described in the section *Administration*.
2. The topic *List and Select Workplan* is described in *List the Available Workplans in the Workplan Library* in the section *Administration*.

# 14 Wizards


- Step 1 - Start the Procedure ..... 106
- Step 2 - Unload/Load/Scan Objects into/from Work Files ..... 107
- Step 3 - Set Parameters ..... 109
- Step 4 - Select Objects ..... 110
- Step 5 - Execute Processing ..... 111
- Step 6 - Continue Processing ..... 112

The Object Handler provides wizards that determine the processing sequence for the following:

- Unloading data from the Natural system environment into Natural work files.
- Loading data from work files into the Natural system environment.
- Scanning the contents of Natural work files.

### To activate the wizards

- In the **Main Menu**, select the **Advanced user** field if required (the field is not selected by default).

The wizards provide the keys PF8 and PF7 to navigate between the screens (steps). Use PF12 to cancel the processing sequence.

The steps described in this section show the processing sequence performed with the unload, load or scan wizard.

## Step 1 - Start the Procedure

### To start the unload, load or scan procedure

1 From the **Main Menu**, choose **Unload**, **Load** or **Scan** by entering any single character next to the function required or by using the corresponding PF key.

The initial **Wizard** screen appears with the following options:

- **Unload/Load/Scan objects into/from Natural work file(s)**.
- **Start Object Handler command procedure**.

2 If you want to unload objects into a work file, load object from a work file or scan objects in a work file, proceed with *Step 2 - Unload/Load/Scan Objects into/from Work Files* below.

Or:

If you want to use a command procedure for unloading, loading or scanning objects, choose **Start Object Handler command procedure** and proceed as follows:

1. On the initial **Wizard** screen, choose **Start Object Handler command procedure**. The **Procedure** screen appears.
2. In the **Name** field, enter the name of a Workplan of the type PROCEDURE by using either of the following options:

- Type in the name of a Workplan of the type PROCEDURE (see also *Workplans*) that should be used for the transaction.
- Choose **Select Workplan** or choose PF5 to display a list of available Workplans of the type PROCEDURE. In the line next to the Workplan you want to select, enter the command `S` or `SE`. Choose ENTER to execute the command and fill the **Name** field on the **Procedure** screen.

3. Select **List Workplan** or choose PF4 if you want to display the Workplan specified.

   See also *List the Available Workplans in the Workplan Library* in the section *Administration*.

3 Choose ENTER to continue.

4 Proceed with *Step 5 - Execute Processing*.

## Step 2 - Unload/Load/Scan Objects into/from Work Files

**To unload, load or scan objects into/from Natural work files**

1 On the initial **Wizard** screen, choose **Unload/Load/Scan objects into/from Natural work file(s)**.

2 Choose ENTER or choose PF8 (Next) to continue. The **Options** screen of the wizard appears with the following fields, commands and alternative PF keys:

| Field | PF Key | Explanation |
|---|---|---|
| **Transfer format** | | Only valid if **Use default options** (this is the default) has been selected.<br><br>If selected, the data to be processed is written in Transfer format to/from the work file. See also *Work File Format* in *Work Files*.<br><br>Unload function:<br>The data to be unloaded is written in Transfer format to the work file. Note that if you want to change the setting of this field for a subsequent unload, you need to return to the **Main Menu** or enter the command `GO UNLOAD END` (see *Commands for Navigation and Special Functions* in *Direct Commands*) and restart the unload function.<br><br>Load and scan functions:<br><br>The data to be loaded or scanned is expected to be in Transfer format. |

| Field | PF Key | Explanation |
|---|---|---|
| **Unicode work file** | | Only applies to the unload function and if **Transfer format** has been selected.<br><br>If this option is selected, all object sources are converted to Unicode/UTF-8 (Universal Transformation Format, 8-bit form) before they are written to the work file. |
| **Use PC File** | | Only applies if Entire Connection is installed.<br><br>If selected, the data to be processed is read from or written to an Entire Connection work file. |
| **PC File** | | Only valid if **Use PC File** has been selected.<br><br>The name of the path and the Entire Connection work file to be used. |
| **Use default options** | | Default options are (this is the default). For the options available, see ***Profile Settings*** and ***Set Additional Options*** in *Settings*. |
| **Set additional options** | PF4 | Only valid if **Use default options** has been selected.<br><br>Invokes the **Options** screen of the wizard where you can modify the default settings and enter additional options for the processing sequence. See ***Set Additional Options*** in *Settings*. |
| **Use Option Workplan** | | If selected, a Workplan of the type OPTION is used (see ***Workplans***). |
| **Name** | | Only valid if **Use Option Workplan** has been selected.<br><br>The name of a Workplan of the type OPTION to be used. |
| **List Option Workplan** | PF6 | Only valid if **Use Option Workplan** has been selected.<br><br>Displays the contents of the Workplan specified in the field Name. |
| **Select Option Workplan** | PF5 | Only valid if **Use Option Workplan** has been selected.<br><br>Displays a selection list of available Workplans of the type OPTION (see also ***List the Available Workplans in the Workplan Library*** in *Administration*). |

3 Select any of the options provided and (if necessary) complete the fields to be used for the processing sequence.

4 Choose ENTER or choose PF8 (Next) to continue.

The **Parameters** screen of the wizard appears.

## Step 3 - Set Parameters

### To set parameters for the processing procedure

1 On the **Parameters** screen, select any of the following options and (if necessary) complete the fields to be used for the processing sequence:

| Field | PF Key | Explanation |
|---|---|---|
| **Do not use parameters** | | If selected (default setting), no parameters are set. |
| **Use global parameters** | | If selected, global parameters are used. See ***Set Global Parameters*** in *Settings*. |
| **Set global parameters** | PF4 | Only valid if **Use global parameters** has been selected.<br><br>If selected, the **Parameters** screen is invoked. See ***Set Global Parameters*** (*Settings*) and `parameter-setting` (*Direct Commands*) for descriptions of keywords and valid input values. |
| **Use Parameter Workplan** | | If selected, a Workplan of the type PARAMETER is used (see ***Workplans***). |
| **Name** | | Only valid if **Use Parameter Workplan** has been selected.<br><br>The name of a Workplan of the type PARAMETER to be used. |
| **List Parameter Workplan** | PF6 | Only valid if **Use Parameter Workplan** has been selected.<br><br>If selected, the contents of the Workplan specified in the field **Name** is displayed. |
| **Select Parameter Workplan** | PF5 | Only valid if the field **Use Parameter Workplan** has been selected.<br><br>If selected, a selection list of available Workplans of the type PARAMETER is displayed (see ***List the Available Workplans in the Workplan Library*** in *Administration*). |

2 Choose ENTER or choose PF8 (Next) to continue.

The **Select Unload/Load/Scan Type** screen appears.

## Step 4 - Select Objects

### To select the type of object you want to process

1 On the **Select Unload/Load/Scan Type** screen, choose one of the three options described below. Note that the first option only applies to the load and scan functions. For the keywords and valid values that apply to each object type, see the relevant explanations in the section *Object Specification*.

1. Select **Load/Scan all objects** to process all objects from the work file.
2. Select a particular type of object:
   - **Natural library objects**
   - **Natural system error messages**
   - **Natural command processor sources**
   - **Natural-related objects**
   - **DDMs**
   - **FDTs**

   Choose ENTER or choose PF8 (Next) to continue.

   Depending on the type of object selected, a screen appears where you can specify selection criteria for the objects to be processed.

   Enter selection criteria and choose **Details** (if available) for further object specifications, if required. For information on **Details**, see the relevant explanation in the section *Object Specification*.
3. Select **Use Selection or List Workplan** to use a Workplan of the type SELECTION or LIST. See also *Workplans*.

   Choose ENTER or choose PF8 (Next) to continue.

   The **Selection or List** screen appears. In the **Name** field, enter the name of a Workplan of the type SELECTION or LIST by using either of the following options:

   - Type in the name of a Workplan.

     Or:

     Choose **Select Workplan** or PF5 (SelWP) to display a list of all Workplans available. In the line next to the Workplan you want to select, enter either the command S or SE.

Choose ENTER to execute the command and fill the **Name** field on the **Selection or List** screen. See also ***List the Available Workplans in the Workplan Library*** in the section *Administration*.

Choose **List Workplan** or PF4 (Li-WP) if you want to display the contents of the Workplan entered in the **Name** field.

2 Choose ENTER or PF8 (Next) to continue.

The wizard displays the processing command generated from the input data.

You can save the command displayed as a Workplan of the type PROCEDURE (see also ***Workplans***), by entering the command SAVE or by choosing PF5 (Save).

## Step 5 - Execute Processing

### To execute the processing procedure

1 On the command execution screen, choose ENTER or choose PF8 (Next) to confirm the settings and to process the objects specified.

If required, choose PF7 (Back) and modify the processing settings before you confirm command execution.

The Object Handler performs the function and displays a confirmation message.

2 Choose ENTER to continue.

A report screen appears with a list of the objects processed.

3 Choose PF3 (Exit) to leave the report screen or choose PF12 (Canc) to terminate the function.

A window appears where you can choose whether to continue processing data.

4 Choose **No** and then ENTER to terminate the function.

Or:

Choose PF12 to terminate the function.

The **Main Menu** appears.

## Step 6 - Continue Processing

### To continue processing

1 On the report screen, choose PF3 (Exit).

A window appears where you choose whether to proceed with the next processing step.

2 Choose **Yes**.

A screen appears with the option to reuse or change previous settings.

# 15 Advanced User


- Activating Advanced User ..... 114
- Processing Objects ..... 114

This section describes how to invoke advanced-user mode and perform the unload, load and scan functions.

**Note:** This parameter is no longer used and is kept for compatibility reasons only.

## Activating Advanced User

### To activate advanced-user mode

- From the **Main Menu**, select the **Advanced user** field (the field is not selected by default).

  Or:

  Set advanced-user mode as the default by specifying the `Advanced-Mode` parameter in your Object Handler profile (see *Profile Settings*).

## Processing Objects

### To process objects in advanced-user mode

1. In the **Main Menu**, check the **Advanced user** field and select **Unload**, **Load** or **Scan**.
2. Choose ENTER to continue.

   The **Unload/Load/Scan Settings** screen appears with the sections **Options** and **Parameters**.
3. Set the options and parameters as described in the section *Settings*.
4. Choose ENTER to continue.

   The **Select Unload/Load/Scan Type** screen appears.
5. Select the objects you want to process: see also the section *Object Specification*.
6. Choose **Details** to specify additional selection criteria: see the relevant sections in *Object Specification*.
7. Choose ENTER to continue.

   - If the parameter `Display-Cmd-in-Advanced-Mode` is set to `N` (No) in the Object Handler profile (this is the default), or if no such profile exists, the command generated from the input data is executed immediately after you have specified the selection data. See also *Profile Settings*.

The **Display Unload/Load/Scan Report** screen appears with a list of the objects processed if the field **Write report** was selected (this is the default). See also ***Work File Options*** in the section *Settings*.

- If the parameter `Display-Cmd-in-Advanced-Mode` is set to `Y` (Yes) in the Object Handler profile (see ***Profile Settings***), or if the command `SET ADVANCEDCMD ON` (see *Commands for Navigation and Special Functions*) was executed earlier, a screen appears, which displays the command generated from the input data.

  You can save the command displayed as a Workplan of the type PROCEDURE (see also ***Workplans***), by entering the command `SAVE` or by choosing PF5 (Save).

  Choose ENTER to confirm command execution or choose PF3 (Exit) to modify the processing settings before confirming command execution.

  The **Display Unload/Load/Scan Report** screen appears with a list of the objects processed if the field **Write report** was selected (this is the default). See also ***Work File Options*** in the section *Settings*.

# 16 Compact Mode


- How to Select Compact Mode ..... 118
- How Instructions are Processed in Compact Mode ..... 118

By selecting compact mode, the advanced user can specify object parameters to execute unload, load or scan functions in only two steps. Other object parameters can also be specified or modified by selecting an appropriate submenu. Another feature of compact mode is to execute one or more processing steps described in wizard mode (see ***Wizard***) without having to pass the entire sequence of processing steps. Compact mode is thus more powerful than advanced mode and requires yet more expert knowledge. This chapter covers the following topics:

## How to Select Compact Mode

There are two ways to select compact mode:

- **Adjust the Object Handler profile**
  See the documentation on the `INPUT-MODE` parameter in *Profile Settings*.
- **By the command** `SET INPUT-MODE C` **or**`SET IM C`
  See the documentation on the `SET INPUT` command in *Commands for Navigation and Special Functions* in *Direct Commands*.

## How Instructions are Processed in Compact Mode

In compact mode, processing instructions can be carried out in two steps. In step one, you choose a function to be executed from the compact mode main menu. As a result, a second menu specific to your choice is displayed. Here, you can enter parameters specific to the function chosen in step one. This section covers both menus:

- Compact Mode Main Menu
- Compact Mode Subsequent Menu

### Compact Mode Main Menu

In the compact mode main menu, the following fields can be specified:

| Field | PF Key | Explanation |
|---|---|---|
| **Function** | | Choose a function:<br><br>U unload (default)<br>L load<br>S scan |

| Field | PF Key | Explanation |
|---|---|---|
| | | **Caution:** Once the unload function has been started, further modifications for **Function** are not permitted unless you terminate the unload function by exiting the menu and restart compact mode. |
| **Object** | | Type of object to be processed:<br><br>L Natural library object (default)<br>E Natural system error message<br>C Natural command processor source<br>R Natural-related object<br>D DDM<br>F FDT<br>A Any Natural object. Only available if `L` or `S` has been selected for **Function.** |
| **Work file format** | | File format for loading and unloading operations to be applied to data to be processed:<br><br>I Internal format (default)<br>T Transfer format<br>See *Work File Format*. |
| **Use PC file** | | Only applies if Entire Connection is installed.<br><br>If selected, the data to be processed is read from or written to an Entire Connection work file. |
| **PC file** | | Only applies if Entire Connection is installed.<br><br>The complete path name to the Entire Connection work file. If your system environment does not accept a backslash (\) separator, use a slash (/) instead. |
| **Write report** | | If set to `Y` writes a report of the objects processed to the report text member specified in the**Report text member** field. The **Write report** option is set to `Y` by default.<br><br>To display the report, enter the internal command `SHOW REPORT FILE` (see *Commands for Navigation and Special Functions* in *Direct Commands*). |
| **Report text member** | | Only valid if **Write report** has been selected. The name of the text member stored in the Workplan library to which the report is written. |
| **Set additional options** | PF4 | If set to "Y" invokes the **Options** screen where you can modify the default settings and enter additional options for the processing sequence. For the options available, see*Set Additional Options*. Cannot be used / Option specifications are ignored when **Use Option Workplan** is set to 'Y'. Note: If any Options have been defined, the text '(Options are defined) ' is displayed. |
| **Use Option Workplan** | | If set to `Y`, a Workplan of type `OPTION` is used. See also *Workplans* |

| Field | PF Key | Explanation |
|---|---|---|
| **Option Workplan name** | | The name of a Workplan of type `OPTION` to be used. |
| | PF5 | Display a selection list of available Workplans of type `OPTION`. See also ***List the Available Workplans in the Workplan Library*** in *Administration*. |
| | PF6 | Displays the contents of the Workplan specified in the **Option Workplan name** field. |
| **Set global parameters** | PF7 | Invokes the **Parameters** screen. See ***Set Global Parameters*** and ***parameter-setting*** (*Direct Commands*) for descriptions of keywords and valid input values.<br><br>Cannot be used / Parameter specifications are ignored when **Use Parameter Workplan** is set to `Y`.<br>Note: If any Parameters have been defined, the text '(`Parameters are defined`) ' is displayed. |
| **Use Parameter Workplan** | | If set to `Y`, a Workplan of the type PARAMETER is used. See also ***Workplans***. |
| **Parameter Workplan name** | | The name of the `PARAMETER` Workplan to be used. |
| | PF8 | Displays a selection list of available Workplans of the type `PARAMETER`. See also ***List the Available Workplans in the Workplan Library*** in *Administration*. |
| | PF9 | Display the contents of the Workplan specified in **Parameter Workplan name**. |

### Compact Mode Subsequent Menu

In the compact mode subsequent menu, you add information specific to the selection you made in the compact mode main menu. For a documentation on attributes to be specified, see ***Object Specification***.

# 17 Restart Load

You can use the restart load function to resume load functions that terminated abnormally. If the load function terminates before the work file has been processed completely, with the restart load you can continue from the point of termination.

The restart load requires that restart information is written to a text member (Natural object of the type Text) in accordance with the selection criteria, options and parameter settings specified for the load.

### To set up the environment during the load

1 On the **Load Options** screen:

- Mark the **Write restart information** option.
- In the **Restart text member** field, enter the name of the text member to which the restart information data is written. The text member must be contained in the Workplan library.

  Or:

  Mark the **Select text member** field next to **Restart text member** and select a text member from the list of Workplans contained in the Workplan library.

The **Load Options** screen is described in *Work File and Report Options* in the section *Settings*.

2 Execute the load function.

### To execute the restart load after an interrupted load

- In the Command line of the Object Handler screen, enter the following command:

```
GO RESTART
```

The **Restart Options** screen appears, where you can specify a text member by entering a name in the **Restart text member** field or by marking **Select text member** and selecting a text member from a list.

Or:

Use the following direct command:

```
RESTART
```

The syntax of `RESTART` is shown in the section ***Basic Command Syntax***.

**Related Topics:**

***Change the Workplan Library*** in the section *Administration*.
`GO RESTART` in the section ***Commands for Navigation and Special Functions***.

# 18 View


- Natural Library Objects ..... 124
- Natural System Error Messages ..... 125
- Natural Command Processor Sources ..... 126
- FDTs ..... 127
- Natural-Related Objects ..... 127
- DDMs ..... 128

This function is used to view all objects contained in your Natural system environment. Depending on the type of object selected, you can also use this function to delete an object if required.

### To invoke the view function

- In the **Main Menu**, choose **View**.

  Or:

  On any other Object Handler screen, enter the following direct command:

  ```
  GO VIEW
  ```

  (See also *Commands for Navigation and Special Functions* in the section *Direct Commands*.)

  The **Select View Type** screen appears with all types of object available for selection.

This section describes how to view the object types listed on the **Select View Type** screen:

## Natural Library Objects

Natural library objects are programming objects and user-defined error messages.

### To view Natural library objects

1. On the **Select View Type** screen, select **Natural library objects**.

   The **View System Files** screen appears with a list of all system files available in the current Natural environment.

   For explanations of the screen columns, see the **Select System File** screen with identical columns, which are described in *Select System File*.

2. In the **Cmd** column, enter any single character next to the system file you want to select. The current FUSER or FNAT system file is selected by default.

   The **View Libraries** screen appears with a list of all libraries available in the system file specified.

   You can start the list of libraries from a particular library, or filter objects by entering a library name or a range of names in the **Library** field. For valid name ranges, see *Name* in the section *Name, Date and Time Specification*.

3. In the **Cmd** column, next to the library you want to select, enter one of the following line commands:

   ```
   L
   ```

```
LI
```

```
S
```

```
SE
```

The **View Library Objects** screen appears with a list of all objects contained in the library specified.

For explanations of this screen, see the description of the **List** screen, which has identical **columns** described in *Select Objects*.

4 In the **Cmd** column, next to the object you want to view, enter either of the following line commands:

```
L
```

```
LI
```

Or:

If required, next to an object you want to delete, enter the following line command:

```
DE
```

Depending on the command entered, either the source code of the object selected is displayed on the screen or a confirmation window appears, which is used to execute the delete function.

## Natural System Error Messages

### To view Natural system error messages

1 On the **Select View Type** screen, select **Natural system error messages**.

The **View System Error Messages** screen appears.

2 If the system error messages required are not stored in the current FNAT or FUSER system file, replace the database ID in the **DBID** field and the file number in the **FNR** field. If required, specify the Adabas password (**Password**) and the cipher code (**Cipher**) for the system file.

If required, specify selection criteria in the fields **Error number range**, **Languages** and **Short/Long/All**. These fields correspond to the **columns** **Number**, **Language** and **S/L** respectively that appear when the view function is executed and the list of error messages is displayed on the **View System Error Messages** screen.

For explanations of this **View System Error Messages** screen, see the description of the **List System Error Messages** screen, which has identical **columns**.

3 When the list of the system error messages selected is displayed on the screen, in the **Cmd** column, next to the error message you want to view, enter either of the following line commands:

```
L
```

```
LI
```

Or:

You can delete an error message by entering the following line command in the **Cmd** column, next to the object required:

```
DE
```

Depending on the command entered, either the source code of the error message selected is displayed on the screen or a confirmation window appears, which is used to execute the delete function.

## Natural Command Processor Sources

### To view Natural command processor sources stored in an Adabas file

1 On the **Select View Type** screen, select **Natural command processor sources**.

The **View Natural Command Processors** screen appears.

2 If the Natural command processor sources required are not stored in the current FUSER system file, enter the required database ID in the **DBID** field and the file number in the **FNR** field.

If required, enter an Adabas password in the **Password** field and a cipher code in the **Cipher** field.

The **View Libraries** screen appears with a list of all libraries where Natural command processor sources are stored.

3 You can start the list of libraries from a particular library, or filter Natural command processor sources by entering a library name or a range of names in the **Library** field. For valid name ranges, see ***Name*** in the section *Name, Date and Time Specification*.

4 In the **Cmd** column, next to the library you want to select, enter one of the following line commands:

```
L
```

```
LI
```

```
S
```

```
SE
```

The **View Command Processors** screen appears with a list of all Natural command processor sources contained in the library specified.

For explanations of this screen, see the description of the **List** screen, which has identical **columns**.

5 You can delete an object by entering the following line command in the **Cmd** column, next to the object required:

```
DE
```

A confirmation window appears, which is used to execute the delete function.

## FDTs

### To view the FDTs available in an Adabas database

1 On the **Select View Type** screen, select **FDTs**.

The **View FDTs** screen appears.

2 If the objects required are not stored in the current FNAT or FUSER system file, replace the database ID in the **DBID** field and, if required, the range of file numbers entered in the **FNR from** and **FNR to** fields.

The **View FDTs for DBID** screen appears with a list of all FDTs in the file range and for the database specified.

## Natural-Related Objects

### To view Natural-related objects

1 On the **Select View Type** screen, select **Natural-related objects**.

The **Select Natural-Related Type** screen appears.

2 Select the Natural-related type of object you want to view: profile, debug environment or DL/I subfile.

A **View** screen appears where you can specify the location of the Natural-related objects you want to view.

3 If the objects required are not stored in the current system file (FNAT for profiles, FUSER for debug environments, FDIC for DL/I subfiles), enter the required database ID in the **DBID** field and a file number in the **FNR** field.

   If required, enter an Adabas password in the **Password** field and a cipher code in the **Cipher** field.

   A list of all Natural-related objects of the type and system file specified is displayed on the screen.

   For explanations of this screen, see the description of the **List** screen, which has identical **columns**.

4 In the **Cmd** column, next to the object you want to view, enter either of the following line commands:

```
L
```

```
LI
```

   Or:

   You can delete an object by entering the following line command in the **Cmd** column, next to the object required:

```
DE
```

   Depending on the command entered, either the source code of the Natural-related object selected is displayed on the screen or a confirmation window appears, which is used to execute the delete function.

## DDMs

### To view Natural DDMs

1 On the **Select View Type** screen, select **DDMs**.

   The **View DDMs** screen appears.

2 If the DDMs required are not stored in the current FDIC system file, enter the required database ID in the **DBID** field and the file number in the **FNR** field.

   If required, enter an Adabas password in the **Password** field and a cipher code in the **Cipher** field.

   An alphabetical list of all DDMs contained in the specified system files is displayed on the screen.

For explanations of this **View DDMs** screen, see the description of the **List** screen, which has identical **columns**.

3 In the **Cmd** column, next to the DDM you want to view, enter either of the following line commands:

```
L
```

```
LI
```

Or:

You can delete a DDM by entering the following line command in the **Cmd** column, next to the DDM required:

```
DE
```

Depending on the line command entered, either the source code of the DDM selected is displayed on the **List DDM** screen or a confirmation window appears, which is used to execute the delete function.

# 19 Find

This function is used to locate objects in your Natural environment and generate a list of the objects found.

### To invoke the find function

- On any Object Handler screen, in the Command line, enter the following:

```
GO FIND
```

For information on the columns that appear on the report screen generated by the find function, refer to the section ***Object Specification***. For the subcommands provided with `GO FIND`, refer to *Commands for Navigation and Special Functions* in the section *Direct Commands*.

# 20 Administration


- List the Available Workplans in the Workplan Library ..... 134
- Create a New Workplan ..... 136
- Change the Workplan Library ..... 138
- Change the Report Library ..... 139

This function is used to maintain Object Handler Workplans.

For information on Workplans and the syntax that applies, refer to the sections *Workplans* and *Direct Commands*.

This section describes the options provided on the **Administration** screen. Instructions for modifying a Workplan are provided in *List the Available Workplans in the Workplan Library*.

## List the Available Workplans in the Workplan Library

This function is used to list all Workplans contained in the Workplan library and to select a Workplan for further processing such as editing or executing the Workplan.

### To list Workplans

- On the **Administration** screen, select **List the available Workplans in the Workplan library** or choose PF4 (List).

  The **List Workplans** screen appears with a list of all Workplans contained in the Workplan library.

  If the Natural object of the type Text is a Workplan, the type of Workplan and the first 50 bytes of the Workplan description are listed. You can choose PF5 to display additional information.

The **List Workplans** screen is also invoked with the select function, which is provided, for example, on the **Unload/Load/Scan Settings** screen.

The columns displayed on the **List Workplans** screen and the commands that can be executed on a Workplan are described in the following section.

- Columns and Commands on the List Workplans screen

### Columns and Commands on the List Workplans screen

The columns and commands provided on the **List Workplans** screen are explained in the following table.

You can use the input fields below each column heading to start the list from a particular Workplan or filter Workplans. Valid input values are mentioned in the table below.

| Column | PF Key | Explanation |
|---|---|---|
| **Cmd** | | The following line commands can be entered in the input field next to the Workplan required:<br><br>`C` or `CH` — Checks the syntax. Only applies to Workplans of the types PROCEDURE, SELECTION, PARAMETER and OPTION.<br>`DE` — Deletes the Workplan.<br>`ED` — Edits the Workplan. You can modify the name of a Workplan or its description in the **Save Workplan** window described in *Saving a Workplan*.<br>`EX` — Executes the Workplan. Only applies to Workplans of the type PROCEDURE.<br>`L` or `LI` — Lists the Workplan.<br>`S` or `SE` — Selects the Workplan to be used for the current function. Only applies if the **List Workplans** screen is invoked with the select function, for example, from the **Unload/Load/Scan Settings** screen. |
| **Name** | | The name of the Workplan.<br><br>You can enter a name or a range of names as described in *Name* in *Name, Date and Time Specification*. |
| **Type** | | The type of Workplan such as PROCEDURE.<br><br>Valid input values are:<br><br>`PROCEDURE` or `P`<br>`SELECTION` or `S`<br>`LIST` or `L`<br>`PARAMETER` or `A`<br>`OPTION` or `O`<br>`TEXT` or `T`<br><br>You can also enter an asterisk (*) for all types, or any combination of the short types, for example `SL`. |
| **Description** | | The description of the Workplan.<br><br>You can enter a description or a range of descriptions as described in *Name* in *Name, Date and Time Specification*. |
| **User ID** | | Only displayed with PF5.<br><br>The ID of the user who created the Workplan. |

| Column | PF Key | Explanation |
|---|---|---|
| | | You can enter a user ID or a range of user IDs as described in ***Name***. |
| **Date** | | Only displayed with PF5.<br><br>The date when the Workplan was created.<br><br>You can enter a date or a range of dates as described in ***Date*** in *Name, Date and Time Specification*. |
| **Time** | | Only displayed with PF5.<br><br>The time when the Workplan was created.<br><br>You can enter a time or a range of times as described in ***Time*** in *Name, Date and Time Specification*. |
| | PF4 | Switches from the additional information display (PF5) to the standard display. |
| | PF5 | Displays additional information: **user ID**, **date** and **time**. |

## Create a New Workplan

This function invokes the **Create a new Workplan** screen where you can specify the type of the new Workplan and the format to be used for editing the Workplan.

If you do not select the **Free Format Editing** option (field not marked; this is the default setting), for Workplans of the types OPTION, PARAMETER and SELECTION, screens with input fields are provided.

If you select the **Free Format Editing** option (field marked) or if you create a Workplan of another type, you will obtain a map with an edit area where you can enter the contents of the Workplan; see also ***Contents of Workplans*** in the section *Workplans*.

For alternative direct command that can be used to set free format editing on and off, see the command `SET` in *Commands for Navigation and Special Functions* in the section *Direct Commands*.

This section covers the following topics:

- Creating a PROCEDURE Workplan
- Creating a LIST Workplan

- Saving a Workplan

## Creating a PROCEDURE Workplan

You can create a Workplan of the type PROCEDURE from the command generated for the current Object Handler function.

### To create a PROCEDURE Workplan from a generated command

1 Execute the function you want to use for the Workplan with an Object Handler wizard until the command generated for the function to be executed is displayed on the screen.

Or:

In advanced-user mode, activate the display of the generated command by choosing either of the following methods:

- Enter the following Object Handler command:

```
SET ADVANCEDCMD ON
```

Or:
In the Object Handler profile, set the parameter `Display-Cmd-in-Advanced-Mode` to `Y` (Yes). For details, see *Profile Settings*.

- Execute the function you want to use for the Workplan until the command generated for the function to be executed is displayed on the screen.

2 Choose PF5 (Save).

The **Save Workplan** window appears.

3 Enter name and description of the new Workplan and choose ENTER.

The Workplan is saved as a PROCEDURE Workplan in the Workplan library. It contains the command generated for the current function.

## Creating a LIST Workplan

For details on creating a Workplan of the type LIST, refer to the section *Object List - LIST Workplan*.

### Saving a Workplan

#### To save a Workplan

1 When you have finished editing a Workplan, in the Command line, enter the following command:

```
SAVE
```

Or:

Choose PF5 (Save).

The **Save Workplan** window appears.

2 Enter or modify name and description of the Workplan and choose ENTER.

The Workplan is saved under the specified name in the Workplan library.

## Change the Workplan Library

This function is used to change the Workplan library. All Workplans must be stored in a Workplan library.

**Note:** You can also set the default library for Workplans by specifying the `Workplan-Library` parameter in your Object Handler profile (see *Profile Settings*).

The **Change Workplan Library** screen provides the following fields:

| Field | Explanation | |
|---|---|---|
| **Library** | The name of the Workplan library. Default is the library `WORKPLAN`. | |
| **Select library** | Displays a list of all Workplan libraries available: see also *Select Library*.<br><br>Equivalent PF key: PF4 (SeLib) | |
| **DBID/FNR** | Specifies the database ID (DBID) and file number (FNR) where the Workplan library is located. If no values are specified, the current FUSER or FNAT system file is used. | |
| **Passw./Ciph.** | The password and cipher code of the Adabas file where the Workplan library is located. | |
| **Store values in profile** | Determines whether the values specified for the Workplan library are stored in the Object Handler profile: | |
| | `N` | Do not store the specified values.<br><br>This is the default. |
| | `U` | Store the specified values in the user-specific profile settings. |

| Field | Explanation | |
|---|---|---|
| | G | Store the specified values in the general profile settings. |
| | See also *Profile Settings*. | |

## Change the Report Library

This function is used to change the report library. All reports that are created when the `REPORT` or `NEWREPORT` option is specified in an Object Handler command are stored in the report library. If no report library is specified, the values specified for the Workplan library are used instead.

**Tip:** You can set the default library for reports by specifying the `Report-Library` parameter in your Object Handler profile (see *Profile Parameters* in the section *Profile Settings*).

The **Change the Report library** function opens the **Change Report Library** screen which provides the following fields:

| Field | Explanation | |
|---|---|---|
| **Library** | The name of the report library. Default is the library `WORKPLAN`. | |
| **Select library** | Displays a list of all libraries available: see also *Select Library*.<br><br>Equivalent PF key: PF4 (SeLib) | |
| **DBID/FNR** | Specifies the database ID (DBID) and file number (FNR) where the report library is located. If no values are specified, the current FUSER or FNAT system file is used. | |
| **Passw./Ciph.** | The password and cipher code of the Adabas file where the report library is located. | |
| **Store values in profile** | Determines whether the values specified for the report library are stored in the Object Handler profile: | |
| | N | Do not store the specified values.<br><br>This is the default. |
| | U | Store the specified values in the user-specific profile settings. |
| | G | Store the specified values in the general profile settings. |
| | See also *Profile Settings*. | |

# 21 Select System File

You can select the system files to be used for the unload function from a list.

You can produce this selection list from an object specification screen of the unload function when performed in advanced-user mode.

The instructions below are an example of using the function when unloading Natural library objects.

### To select a system file from a list

1 On the **Unload Natural Library Objects** screen, choose **Select DBID/FNR** or choose PF5 (DBIDs).

The **Select System File** window appears where the system files available in the current Natural environment are listed with their names (**System File**), database IDs (**DBID**) and file numbers (**FNR**). `User defined` denotes a system file specified by the user.

2 Select the system file you want to use for function processing by entering any single character in the **Sel** column next to the system file required. The FUSER system file is selected by default.

Or:
In the **DBID** and **FNR** fields next to `User-defined`, you can enter the database ID and file number of the system file you want to select. If required, in the **Password/VSAM name** and **Cipher Code** columns, enter the Adabas password or VSAM name and the Adabas cipher code for the system file.

The database ID and file number of the system file selected are entered in the **DBID/FNR** fields of the **Unload Natural Library Objects** screen. If you selected the default system file, these fields remain empty.

# 22 Select Library

You can select the library to be used for the unload function from a list.

The selection list is produced with the **Select library** (or **Select**) function, which is provided on the object specification screen of the unload function when performed in advanced-user mode.

The instructions below are examples of selecting single or multiple libraries when unloading Natural library objects.

### To select a single library from a list

1 On the **Unload Natural Library Objects** screen, choose **Select library** or PF4 (SeLib).

   The **Select Library** window appears with a list of all libraries and the database IDs (**DBID**) and file numbers (**FNR**) of the system file where the libraries are stored.

2 In the **Cmd** column, next to the library required, enter any single character.

3 Choose ENTER.

   The **Library** field and the **DBID/FNR** fields of the **Unload Natural Library Objects** screen are filled with the specified name and numbers respectively. If no values (or 0) are entered in the **DBID/FNR** fields, the current FUSER and FNAT system files are selected.

### To list and select multiple libraries

1 On the **Unload Natural Library Objects** screen, choose **Select library** or PF4 (SeLib).

   The **Select Library** window appears with a list of all libraries and the database IDs (**DBID**) and file numbers (**FNR**) of the system file where the libraries are stored.

2 In the **Library** field, enter a name or a range of names to filter the libraries you want to select. If you enter a single library name, the list will start with this library. For valid name ranges, see ***Name*** in the section *Name, Date and Time Specification*.

Or:

In the **DBID** and **FNR** fields, enter the database ID and file number of the system file that contains the libraries you want to select. If no values (or 0) are entered, the current FUSER and FNAT system files are used.

Note that **DBID** and **FNR** are read-only fields when the **Select Library** window has been invoked from an **Exceptions** screen.

3 Choose ENTER.

The **Select Library** window now lists all libraries of the specified range.

4 Choose PF4 (Se Rng).

The **Library** field and the **DBID/FNR** fields of the **Unload Natural Library Objects** screen are filled with the specified name (or range) and numbers respectively. If no values (or 0) are entered in the **DBID/FNR** fields, the current FUSER and FNAT system files are selected.

# 23 Select System Error Messages


- Columns and Commands ..... 146

You can select the Natural system error messages to be unloaded from a list.

You can produce this selection list from an object specification screen of the unload function when performed in advanced-user mode.

**To select Natural System error messages**

- On the **Unload Natural System Error Messages** screen, if required, change the message numbers in the **Error number from/to** fields (default is the full range of numbers) and select **Select system error messages**.

  The **List System Error Messages** screen appears with a list of all system error messages contained in the system file specified.

  This screen is described in the following section.

**Note:** The select function for user-defined error messages is described in the section *Select Objects*.

## Columns and Commands

The columns and commands provided on the **List System Error Messages** screen are explained in the following table.

You can use the input fields below each column heading to start the list from a particular system error message or filter messages. Valid input values are mentioned in the table below.

| Column | PF Key | Explanation | |
|---|---|---|---|
| **Cmd** | | One of following line commands can be entered in the input field next to the system error message required: | |
| | | | |
| | | L or LI | Lists the short and long texts of the message. |
| | | | |
| | | S or SE<br>UL or U | Selects the message for subsequent unloading.<br>**Attention:**<br>Any of these commands only marks the message selected for subsequent processing. To execute the unload function, you need to choose PF2 (Unloa) described below. |
| | | | |
| | | DE | Deletes the message. |
| | | | |

| Column | PF Key | Explanation |
|---|---|---|
| | | DL — Only deletes the long text of the message. |
| **Number** | | The number of the system error message.<br><br>You can enter a number or a range of numbers. Valid ranges are:<br><br>*value** — All messages with numbers that begin with *value*.<br>*value*> — All messages with numbers greater than or equal to *value*. Example: 10><br>*value*< — All messages with numbers less than or equal to *value*. Example: 100< |
| **S/L** | | The kind of system error message text:<br><br>S — Short text.<br>L — Long text.<br>A — Short and/or long text. |
| **Language** | | The language code of the system error message.<br>You can enter up to 8 valid language codes (for example, 1 for English) for the error messages to be selected.<br><br>An asterisk (*) selects all language codes. |
| **Error Message Text** | | The short text of the system error message. |
| | PF2 | Starts unloading the system error messages selected for processing.<br><br>As an alternative, in the Command line, you can enter either of the following direct commands:<br><br>UNLOAD<br>or<br>UNLD |
| | PF11 | Marks all system error messages listed for subsequent unloading with **PF2**.<br><br>As an alternative, in the Command line, you can enter either of the following direct commands:<br><br>SELECT ALL<br>or<br>SEL ALL |

# 24 Select Objects


- Columns and Commands on List Screens ..... 150

You can select the objects to be unloaded from a list. This selection list can also be used for other purposes such as listing the source of an object or deleting it.

The selection list is produced with the **Select objects** (or **Select**) function, which is provided on the object specification screen of the unload function when performed in advanced-user mode.

The selection list is displayed on a **List** screen, which is described in the following section.

**Note:** The select function for Natural system error messages is described in the section *Select System Error Messages*.

## Columns and Commands on List Screens

The columns and commands provided on a **List** screen are explained in the following table.

The display of the columns contained on a **List** screen depends on the type of object selected from the **Select Unload Type** menu. The type of object processed is contained in the screen title, for example, **List Library Objects** or **List Command Processors**.

You can use the input fields below each column heading to start the list from a particular object or filter objects. Valid input values are mentioned in the table below.

| Column | PF Key | Explanation | |
|---|---|---|---|
| **Cmd** | | One of following line commands can be entered in the input field next to the object required: | |
| | | | |
| | | `L` or `LI` | Lists the source code of the object (not applicable to Natural command processor sources).<br><br>For a user-defined error message: lists the short and the long texts of the error message. |
| | | | |
| | | `S` or `SE`<br>`UL` or `U` | Selects the object for subsequent unloading.<br><br>**Attention:**<br>Any of these commands only marks the object selected for subsequent processing. To execute the unload function, you need to choose **PF2** (Unloa) described below. |
| | | | |
| | | `DE` | Deletes the object. |
| **Name** | | The object name.<br><br>You can enter a name or a range of names: see *Name* in *Name, Date and Time Specification*. | |

| Column | PF Key | Explanation |
|---|---|---|
| | | For a user-defined error message, the message number and the language code is displayed. For example: `10 (Lang =1)` denotes message number 10 in language 1 (English). |
| **Type** | | The type of Natural library object such as `Program`.<br><br>Valid input values are one or more object-type codes such as `P` for program. For a list of codes, see `NATTYPE` in the section *`select-clause`*.<br><br>For a Natural-related object of the type profile or DL/I subfile, this column contains the type of profile (such as `Editor`) or DL/I subfile (such as `NBD Subfile`). |
| **S/C** | | The kind of Natural library object: by default, all source (`S`) objects and/or cataloged (`C`) objects available are displayed on the screen.<br><br>Valid input values are one or more of the following codes:<br><br>`S` — Source objects only.<br>`C` — Cataloged objects only.<br>`S/C` — Both source and cataloged objects if both exist.<br>`W` — All STOWed objects: source and cataloged objects with identical date and time.<br>`*` — All source objects and/or cataloged objects.<br><br>For a user-defined error message, this column contains the short text of the error message. |
| **M** | | The programming mode of the Natural library object. By default, any mode is displayed.<br><br>Valid input values are one or more of the following codes:<br><br>`S` Structured mode only.<br>`R` Reporting mode only.<br>`*` Any mode, structured and/or reporting.<br><br>For a user-defined error message, this column contains the short text of the error message. |
| **Version** | | The Natural version under which the Natural library object was saved and/or cataloged.<br><br>For valid ranges of versions, see ***Name*** in *Name, Date and Time Specification*.<br><br>For a user-defined error message, this column contains the short text of the error message. |
| **User ID** | | The ID of the user who saved or cataloged the Natural library object or DDM.<br>You can enter a single user ID or a range of user IDs: see ***Name***.<br><br>For a user-defined error message, this column contains the short text of the error message. |

| Column | PF Key | Explanation |
|---|---|---|
| **Date** | | The date when the Natural library object or DDM was saved or cataloged. You can enter a date or a range of dates: see ***Date*** in *Name, Date and Time Specification*.<br><br>For a user-defined error message, this column contains the short text of the error message. |
| **Time** | | The time when the Natural library object or DDM was saved or cataloged. You can enter a time or a range of times: see ***Time*** in *Name, Date and Time Specification*.<br><br>For a user-defined error message, this column contains the short text of the error message. |
| **DBID** | | The database ID of the system file where DDMs or Natural-related objects are stored. |
| **FNR** | | The file number of the system file where DDMs or Natural-related objects are stored. |
| | PF2 | Starts unloading the objects selected for processing.<br><br>As an alternative, in the Command line, you can enter either of the following direct commands:<br><br>`UNLOAD`<br>or<br>`UNLD` |
| | PF11 | Marks all objects listed for subsequent unloading with **PF2**.<br><br>As an alternative, in the Command line, enter either of the following direct commands:<br><br>`SELECT ALL`<br>or<br>`SEL ALL` |

# 25 Object Specification

The Object Handler Main Menu provides the **Select Unload/Load/Scan Type** screen where you can select the types of object to be processed or specify a Workplan of the type SELECTION or LIST.

For each type of object selected, you are provided individual object-specification screens. These screens are used to specify selection criteria for the objects to be processed.

**Note:** As a time-saving alternative, advanced users can use compact mode, see *Compact Mode*.

This section describes the options provided on each object-specification screen. If a field or function key (PF key) described in this section only appears with a particular function and/or in advanced-user mode, this is indicated by an appropriate remark such as "Only applies to the unload function in advanced-user mode".


**All Objects on the Work File**

**Natural Library Objects**

**Natural System Error Messages**

**Natural Command Processor Sources**

**Natural-Related Objects**

**DDMs**

**FDTs**

**Use Selection or List Workplan**

# 26 Object Specification - All Objects on the Work File

**Only applies to the load or scan function.**

The option **Load/Scan All Objects on the Work File** is used to select all objects available in the work file for processing. In advanced-user mode, from the **Load/Scan All Objects** screen, you can invoke the **Settings** screen where you can specify option and parameter settings. See the section *Settings*.

# 27 Object Specification - Natural Library Objects


- Natural Library Objects ..... 158
- Natural Library Object Details ..... 159
- Natural Library Object Properties ..... 161
- Natural Library Object Exceptions ..... 162
- Natural Library Object Exception Properties ..... 163

This section describes the options provided on the object-specification screens for processing Natural library objects. Natural library objects are programming objects and user-defined error messages.

For descriptions of keywords and valid input values, see also `select-clause` in the section *Direct Commands*.

## Natural Library Objects

The screen **Unload/Load/Scan Natural Library Objects** provides the following fields and PF keys:

| Field | PF Key | Explanation |
|---|---|---|
| **DBID/FNR** | | Only applies to the unload function.<br><br>The database ID (DBID) and file number (FNR) of the system file where the Natural libraries are stored. If no values (or 0) are specified, the current FUSER or FNAT system file is used. |
| **Select DBID/FNR** | PF5<br><br>(advanced-user mode only) | Only applies to the unload function.<br><br>Displays a selection list of system files available. |
| **Password/Cipher** | | Only applies to the unload function.<br><br>The password and cipher code for the Adabas file where the Natural libraries are stored. |
| **Library** | | The name of a library or a range of names: see ***Name*** in *Name, Date and Time Specification*. |
| **Select library** | PF4 | Displays a selection list of all libraries available. See also ***Select Library***. |
| **Object name** | | The name of a Natural programming object or a range of names: see ***Name***.<br><br>Only evaluated if the field **Natural programming objects** (default setting) is selected on the screen **Natural Library Objects, Details**. See also ***Natural Library Object Details***. |
| **Select objects** | | Only applies to the unload function in advanced-user mode.<br><br>If no library range is specified, a selection list of all Natural objects available is displayed (see also ***Select Objects***). |
| **Error number from/to** | | A valid range (1 - 9999) of user-defined error messages delimited by the first and the last message number.<br><br>Only evaluated if the field **Error messages** (default setting) is selected on the screen **Natural Library Objects, Details** (see also ***Natural Library Object Details***). |

| Field | PF Key | Explanation |
|---|---|---|
| **Details** | PF6 | Invokes the screen **Natural Library Objects, Details** where you can enter more detailed object specifications. See ***Natural Library Object Details***. |
| **Settings** | PF7 | Only applies to functions executed in advanced-user mode.<br><br>Invokes the **Unload/Load/Scan Settings** screen where you can specify option and parameter settings: see ***Settings***. |

## Natural Library Object Details

The screen **Unload/Load/Scan Natural Library Objects, Details** is used to specify further selection criteria for Natural library objects.

For descriptions of keywords and valid input values, see also `select-clause` in the section *Direct Commands*.

The screen **Unload/Load/Scan Natural Library Objects, Details** provides the following fields and PF keys:

| Field | PF Key | Explanation |
|---|---|---|
| **Library** | | The name of a library or a range of names: see ***Name*** in *Name, Date and Time Specification*.<br><br>Ranges are not allowed if the **Use Predict set** option is selected. |
| **Select**<br><br>(Library) | PF4 | Displays a selection list of all libraries available. See also ***Select Library***. |
| **DBID/FNR** | | See **DBID/FNR** in *Natural Library Objects* above. |
| **Passw./Ciph.** | | Only applies to the unload function.<br><br>The password and cipher code of the Adabas file where the Natural libraries are stored. |
| Object Types:<br><br>**Natural programming objects** | | Natural programming objects. |
| Object Types:<br><br>**Error messages** | | User-defined error messages. |
| **Object name** | | See **Object name** in *Natural Library Objects* above. |
| **Use Predict set** | | Only applies to the unload and find functions and if Predict is installed. |

| Field | PF Key | Explanation |
|---|---|---|
| | | This option is used to read the names of the objects to be processed from a retained set. A retained set is created with the save set option of the `LIST XREF` command.<br><br>If the **Use Predict set** option is selected, the following applies:<br><br>■ The **Object name** field must contain asterisk (*) indicating all objects. This is the default setting.<br>■ The **Library** field must contain the name of a single library. Name ranges are not allowed.<br>■ The **Set number** field must be filled.<br><br>For detailed information on Predict sets, refer to the *Predict* documentation. |
| **Set number** | | Only applies if **Use Predict set** is selected.<br><br>A one- or two-digit number that identifies the retained set to be used. |
| **Set library** | | Only applies if **Use Predict set** is selected.<br><br>The name of the library to be searched for a Predict set. If you do not specify a name, the library entered in the **Library** field is used by default. |
| **Set user** | | Only applies if **Use Predict set** is selected.<br><br>The ID of the user who created the retained set. If no ID is entered, the ID specified with the system variable `*USER` (see the *System Variables* documentation) is used. |
| Programming Object Options:<br><br>**S/C-Kind** | | The kind of Natural programming object:<br><br>`S` — Source objects only.<br>`C` — Cataloged objects only.<br>`A` or `*` — All source objects and/or cataloged objects. This is the default setting.<br>`W` — All STOWed objects: source and cataloged objects with identical date and time.<br>`B` — Both source and cataloged objects if both exist.<br><br>**Note:** `W` and `B` are valid for the unload function only. Though `W` and `B` can also be entered for the load or scan function, they are treated like `A`. |
| Programming Object Options:<br><br>**Natural types** | | A Natural object-type code such as `P` for program. For a list of valid codes, see `NATTYPE` in the section *`select-clause`*. |

| Field | PF Key | Explanation |
|---|---|---|
| **Select Natural types** | PF6 | Invokes a window where you can select one or more types of Natural object. |
| **Properties** | PF7 | Invokes an extra screen where you can specify additional properties of Natural programming objects: see ***Natural Library Object Properties***. |
| Error Messages:<br><br>**Error number from/to** | | A range of user-defined error messages as entered in the **Error number from/to** fields (see *Natural Library Objects* above). |
| Error Messages:<br><br>**Language codes** | | Up to 8 valid language codes (for example, code 1 for English) of the specified error messages.<br><br>An asterisk (*) selects all language codes. |
| Error Messages:<br><br>**S/L-Kind** | | The kind of error message text:<br>S — Short text.<br>L — Long text.<br>A — Short and/or long text. This is the default.<br>B — Short and long texts if both exist (unload function only). |
| **Exceptions** | PF8 | Invokes an extra screen where you can specify exceptions to the selection of Natural programming objects: see ***Natural Library Object Exceptions***. |

## Natural Library Object Properties

The screen **Unload/Load/Scan Library Objects, Properties** is used to specify properties for the Natural library objects selected for processing.

For descriptions of keywords and valid input values, see also `select-clause` in the section *Direct Commands*.

The screen **Unload/Load/Scan Library Objects, Properties** provides the following fields:

| Field | Explanation |
|---|---|
| **User ID** | The ID of the user who saved or cataloged a Natural programming object. Specify a single user ID or a range of user IDs: see ***Name*** in *Name, Date and Time Specification*. |
| **Programming mode** | The programming mode of the Natural programming objects:<br>R — Reporting mode only.<br>S — Structured mode only. |

| Field | Explanation |
|---|---|
| | A — No mode check performed. This is the default setting. |
| **Natural version** | The Natural version of the Natural programming objects.<br><br>You can also specify a range of versions: see ***Name***. |
| Object Date:<br><br>**Select all objects (no date check)** | Selects all objects, regardless of their date. |
| Object Date:<br><br>**Select objects modified between/and** | Selects all objects with a save or catalog date and/or time within the range specified in these fields by entering a precise start date and/or time and/or an end date and/or time.<br><br>For valid input values, see ***Date*** and ***Time*** in *Name, Date and Time Specification*. Special dates allowed are: `TODAY`, `YESTERDAY`, `MONTH` and `YEAR`. |
| Object Date:<br><br>**Select objects modified on** | Selects all objects with a save or catalog date and/or time that fits the date/time specified in these fields by entering a precise date and/or time.<br><br>For valid input values, see ***Date*** and ***Time***. Special dates allowed are: `TODAY` and `YESTERDAY`. |
| Object Size:<br><br>**Select all objects (no size check)** | Selects all objects, regardless of their size. |
| Object Size:<br><br>**Select objects with size between/and** | Selects all objects with a size within the range specified in these fields by entering a start size and/or an end size. |
| Object Size:<br><br>**Select objects with size** | Selects all objects with a size that fits the size specified in this field. |

## Natural Library Object Exceptions

The screen **Unload/Load/Scan Library Objects, Exceptions** is used to specify exceptions to the selection of Natural library objects.

All objects that match the selection criteria specified in ***Natural Library Objects***, ***Natural Library Object Details*** and ***Natural Library Object Properties*** are checked against the specifications made on the screen **Unload/Load/Scan Library Objects, Exceptions**. Objects that match *all* specifications defined as exceptions, are exempted from processing.

For descriptions of keywords and valid input values, see also `select-clause` in the section *Direct Commands*.

The screen **Unload/Load/Scan Library Objects, Exceptions** is basically identical to the screen **Unload/Load/Scan Natural Library Objects, Details**. See the relevant section for explanations of the fields, commands and alternative PF keys listed in the table below. The field **Add/change properties for selection** is used to specify additional properties for Natural programming object exceptions: see *Natural Library Object Exception Properties*.

| Field | PF Key |
|---|---|
| **Library** | PF4 |
| **Select**<br><br>(Library) | |
| Object Types:<br><br>**Natural programming objects**<br>**Error messages** | |
| **Object name** | |
| **S/C-Kind** | |
| **Natural types** | |
| **Select Natural types** | PF6 |
| **Properties** | PF7 |
| **Error number** | |
| **S/L-Kind** | |
| **Languages** | |

## Natural Library Object Exception Properties

The screen **Unload/Load/Scan Library Objects, Exceptions** is used to specify exceptions to the properties of the Natural library objects selected for processing.

The screen provides the following fields:

| Field | Explanation |
|---|---|
| **User ID** | See **User ID** in *Natural Library Object Properties*. |
| **Programming mode** | See **Programming mode** in *Natural Library Object Properties*. |
| **Natural version** | See **Natural version** in *Natural Library Object Properties*. |
| Object Date:<br><br>**Ignore object date** | Performs no date check. Objects are processed, regardless of their date. |

| Field | Explanation |
|---|---|
| Object Date:<br><br>**Exclude objects modified between/and** | Exempts from processing all objects with a save or catalog date and/or time within the range specified in these fields by entering a precise start date and/or time and/or an end date and/or time.<br><br>For valid input values, see ***Date*** and ***Time*** in *Name, Date and Time Specification*. Special dates allowed are: `TODAY`, `YESTERDAY`, `MONTH` and `YEAR`. |
| Object Date:<br><br>**Exclude objects modified on** | Exempts from processing all objects with a save or catalog date and/or time that fits the date/time specified in these fields by entering a precise date and/or time.<br><br>For valid input values, see ***Date*** and ***Time***. Special dates allowed are: `TODAY` and `YESTERDAY`. |
| Object Size:<br><br>**Ignore object size** | Performs no size check. Objects are processed, regardless of their size. |
| Object Size:<br><br>**Exclude objects with size between/and** | Exempts from processing all objects with a size within the range specified in these fields by entering a start size and/or an end size. |
| Object Size:<br><br>**Exclude objects with size** | Exempts from processing all objects with a size that fits the size specified in this field. |

# 28 Object Specification - Natural System Error Messages


- Natural System Error Messages ..... 166
- Natural System Error Message Details ..... 166
- Natural System Error Message Exceptions ..... 167

This section describes the options provided on the object-specification screens for processing Natural system error messages from the specified system file.

For descriptions of keywords and valid input values, see also `select-clause` in the section *Direct Commands*.

## Natural System Error Messages

The screen **Unload/Load/Scan Natural System Error Messages** provides the following fields and PF keys:

| Field | PF Key | Explanation |
|---|---|---|
| **Error number from/to** | | A range of Natural system error messages delimited by the first and the last message number. Select **Select system error messages** for a list of all system error messages available. |
| **Details** | PF6 | Invokes the screen **Unload/Load/Scan Natural Library Objects, Details** where you can enter more detailed object specifications: see ***Natural System Error Message Details***. |
| **Settings** | PF7 | Invokes the screen **Unload/Load/Scan Settings** where you can specify option and parameter settings. See ***Settings***. |

## Natural System Error Message Details

The screen **Unload/Load/Scan System Error Messages, Details** is used to specify further selection criteria for Natural system error messages.

For descriptions of keywords and valid input values, see also `select-clause` in the section *Direct Commands*.

The screen **Unload/Load/Scan System Error Messages, Details** provides the following fields and PF keys:

| Field | PF Key | Explanation |
|---|---|---|
| **DBID/FNR** | | Only applies to the unload function.<br><br>The database ID (DBID) and the number of the Adabas (FNR) file where the Natural error messages are stored. |
| **Password/Cipher** | | Only applies to the unload function.<br><br>The password and cipher code for the Adabas file where the Natural error message sources are stored. |

| Field | PF Key | Explanation |
|---|---|---|
| **Error number from/to** | | See **Error number** in *Natural System Error Messages* above. |
| **Language codes** | | See **Language codes** in *Natural Library Object Details* . |
| **S/L-Kind** | | See **S/L-Kind** in *Natural Library Object Details* . |
| **Exceptions** | PF8 | Invokes an extra screen where you can specify exceptions to the selection of Natural system error messages: see ***Natural System Error Message Exceptions***. |

## Natural System Error Message Exceptions

The screen **Unload/Load/Scan System Error Messages, Exceptions** is used to specify exceptions to the selection of Natural system error messages.

All Natural system error messages that match the selection criteria specified in ***Natural System Error Messages*** and ***Natural System Error Message Details*** are checked against the specifications made on the screen **Unload/Load/Scan System Error Messages, Exceptions**. Error messages that match *all* specifications defined as exceptions, are exempted from processing.

For explanations of the fields provided on the exceptions screen, see ***Natural System Error Message Details*** above.

For descriptions of keywords and valid input values, see also `select-clause` in the section *Direct Commands*.

# 29 Object Specification - Natural Command Processors


- Natural Command Processors ..... 170
- Natural Command Processor Source Exceptions ..... 171

This section describes the options provided on the object-specification screens for processing Natural command processor sources.

For descriptions of keywords and valid input values, see also `select-clause` in the section *Direct Commands*.

## Natural Command Processors

The screen **Unload/Load/Scan Natural Command Processors** provides the following fields and PF keys:

| Field | PF Key | Explanation |
|---|---|---|
| **Library** | | The name of a Natural command processor library or a range of names: see ***Name*** in *Name, Date and Time Specification*. |
| **Select library** | PF4 | Invokes a selection list of Natural command processor libraries available. See also ***Select Library***. |
| **DBID/FNR** | | Only applies to the unload function.<br><br>The database ID (DBID) and the file number of the Adabas (FNR) file where the Natural command processor sources are stored. |
| **Password/Cipher** | | Only applies to the unload function.<br><br>The password and cipher code for the Adabas file where the Natural command processor sources are stored. |
| **Object name** | | The name of a Natural command processor source or a range of names: see ***Name***. |
| **Select objects** | | Only applies to the unload function.<br><br>If no library range has been specified and this field is selected, a selection list of Natural command processor sources available is displayed (see also ***Select Objects***). |
| **Exceptions** | PF8 | Invokes an extra screen where you can specify exceptions to the selection of Natural command processor sources: see ***Natural Command Processor Source Exceptions***. |
| **Settings** | PF7 | Invokes the **Unload/Load/Scan Settings** screen where you can specify option and parameter settings. See ***Settings***. |

## Natural Command Processor Source Exceptions

The screen **Unload/Load/Scan Natural Command Processors, Exceptions** is used to specify exceptions to the selection of Natural command processor sources.

All objects that match the selection criteria specified in ***Natural Command Processor Sources*** are checked against the specifications made on the screen **Unload/Load/Scan Natural Command Processors, Exceptions**. Natural command processor sources that match *all* specifications defined as exceptions, are exempted from processing.

For explanations of the fields provided in the exceptions window, see ***Natural Command Processor Sources*** above.

For descriptions of keywords and valid input values, see also `select-clause` in the section *Direct Commands*.

# 30 Object Specification - Natural-Related Objects


- Natural Profiles ..... 174
- Natural Debug Environments ..... 175
- Natural DL/I Subfiles ..... 176

This section describes the options provided on the object-specification screens for processing Natural-related objects. Natural-related objects are profiles, debug environments and DL/I subfiles.

For descriptions of keywords and valid input values, see also `select-clause` in the section *Direct Commands*.

When you select Natural-related objects on the **Unload/Load Type** screen, the **Select Related Type** screen appears where you can specify the type of the Natural-related object: Natural profiles, debug environments or DL/I subfiles.

## Natural Profiles

The screen **Unload/Load/Scan Natural Profiles** provides the following fields and PF keys:

| Field | PF Key | Explanation |
|---|---|---|
| **DBID/FNR** | | Only applies to the unload function.<br><br>The database ID (DBID) and file number (FNR) of the Adabas file where the Natural profiles are stored.<br><br>If no values (or `0`) are specified, the current FNAT system file is used. |
| **Password/Cipher** | | Only applies to the unload function.<br><br>The password and cipher code for the Adabas file where the Natural profiles are stored. |
| **Select**<br><br>(DBID/FNR) | PF5 | Only applies to the unload function.<br><br>Invokes the **Select System File** window with a list of all system files available in your Natural environment: see also ***Select System File***. |
| **Profile Types** | | The type(s) of profile to be processed:<br><br>parameter, editor, map and/or device. |
| **Object name** | | The name of a profile or a range of names: see ***Name*** in *Name, Date and Time Specification*. |
| **Select**<br><br>(Object name) | | Only applies to the unload function.<br><br>Invokes the **List Profiles** screen with a selection list of profiles available (see also ***Select Objects***). |
| **Exceptions** | PF8 | Invokes an extra screen where you can specify exceptions to the selection of profiles: see ***Natural Profile Exceptions***. |
| **Settings** | PF7 | Invokes the **Unload/Load/Scan Settings** screen where you can specify option and parameter settings: see ***Settings***. |

For descriptions of keywords and valid input values, see also `select-clause` in the section *Direct Commands*.

- Natural Profile Exceptions

### Natural Profile Exceptions

The screen **Unload/Load/Scan Natural Profiles, Exceptions** is used to specify exceptions to the selection of Natural profiles.

All objects that match the selection criteria specified in ***Natural Profiles*** are checked against the specifications made on the screen **Unload/Load/Scan Natural Profiles, Exceptions**. Objects that match *all* specifications defined as exceptions, are exempted from processing.

For descriptions of keywords and valid input values, see also `select-clause` in the section *Direct Commands*.

The screen **Unload/Load/Scan Natural Profiles, Exceptions** provides the following fields:

| Field | Explanation |
|---|---|
| **Object name** | The name of a profile or a range of names: see ***Name*** in *Name, Date and Time Specification*. |
| **Profile Types** | The type(s) of profile to be processed:<br><br>parameter, editor, map and/or device. |

## Natural Debug Environments

The screen **Unload/Load/Scan Natural Debug Environments** provides the following fields and PF keys:

| Field | PF Key | Explanation |
|---|---|---|
| **Library** | | The name of a library or a range of names: see ***Name*** in *Name, Date and Time Specification*. |
| **Select library** | PF4 | Displays a selection list of all libraries available. See also ***Select Library***. |
| **DBID/FNR** | | Only applies to the unload function.<br><br>The database ID (DBID) and file number (FNR) of the Adabas file where the Natural debug environments are stored.<br><br>If no values (or `0`) are specified, the current FNAT system file is used. |
| **Password/Cipher** | | Only applies to the unload function.<br><br>The password and cipher code for the Adabas file where the Natural debug environments are stored. |
| **Object name** | | The name of a debug environment or a range of names: see ***Name***. |
| **Select objects** | | Only applies to the unload function. |

| Field | PF Key | Explanation |
|---|---|---|
| | | Displays a selection list of debug environments available (see also ***Select Objects***). |
| **Exceptions** | PF8 | Invokes an extra screen where you can specify exceptions to the selection of profiles: see ***Natural Debug Environment Exceptions***. |
| **Settings** | PF7 | Invokes the **Unload/Load/Scan Settings** screen where you can specify option and parameter settings: see ***Settings***. |

For descriptions of keywords and valid input values, see also `select-clause` in the section *Direct Commands*.

- Natural Debug Environment Exceptions

### Natural Debug Environment Exceptions

The screen **Unload/Load/Scan Debug Environments, Exceptions** is used to specify exceptions to the selection of Natural debug environments.

All objects that match the selection criteria specified in ***Natural Debug Environments*** are checked against the specifications made on the screen **Unload/Load/Scan Debug Environments, Exceptions**. Objects that match `all` specifications defined as exceptions, are exempted from processing.

For descriptions of keywords and valid input values, see also `select-clause` in the section *Direct Commands*.

The screen **Unload/Load/Scan Debug Environments, Exceptions** provides the following fields:

| Field | Explanation |
|---|---|
| **Library** | The name of a library or a range of names: see ***Name*** in *Name, Date and Time Specification*. |
| **Select**<br><br>(Library) | Displays a selection list of all libraries available. See also ***Select Library***. |
| **Object name** | The name of a debug environment or a range of names: see ***Name***. |

## Natural DL/I Subfiles

The screen **Unload/Load/Scan Natural DL/I Subfiles** provides the following fields and PF keys:

| Field | PF Key | Explanation |
|---|---|---|
| **DBID/FNR** | | Only applies to the unload function.<br><br>The database ID (DBID) and file number (FNR) of the Adabas file where the Natural DL/I subfiles are stored.<br><br>If no values (or `0`) are specified, the current FDIC system file is used. |
| **Password/Cipher** | | Only applies to the unload function.<br><br>The password and cipher code for the Adabas file where the Natural DL/I subfiles are stored. |
| **Select**<br><br>(DBID/FNR) | PF5 | Only applies to the unload function.<br><br>Invokes the **Select System File** window with a list of all system files available in your Natural environment: see also ***Select System File***. |
| **Subfile Types** | | The type(s) of DL/I subfile to be processed: NSB and/or NDB. |
| **Object name** | | The name of a DL/I subfile or a range of names: see ***Name*** in *Name, Date and Time Specification*. |
| **Select**<br><br>(Object name) | | Only applies to the unload function.<br><br>Displays a selection list of DL/I subfiles available (see also ***Select Objects***). |
| **Exceptions** | PF8 | Invokes an extra screen where you can specify exceptions to the selection of profiles: see ***Natural DL/I Subfile Exceptions***. |
| **Settings** | PF7 | Invokes the **Unload/Load/Scan Settings** screen where you can specify option and parameter settings: see ***Settings***. |

- Natural DL/I Subfile Exceptions

## Natural DL/I Subfile Exceptions

The screen **Unload/Load/Scan Natural DL/I Subfiles, Exceptions** is used to specify exceptions to the selection of Natural DL/I subfiles.

All objects that match the selection criteria specified in ***Natural DL/I Subfiles*** are checked against the specifications made on the screen **Unload/Load/Scan Natural DL/I Subfiles, Exceptions**. Objects that match `all` specifications defined as exceptions, are exempted from processing.

For descriptions of keywords and valid input values, see also `select-clause` in the section *Direct Commands*.

The screen **Unload/Load/Scan Natural DL/I Subfiles, Exceptions** provides the following fields:

| Field | Explanation |
|---|---|
| **Object name** | The name of a DL/I subfile or a range of names: see ***Name*** in *Name, Date and Time Specification*. |
| **Subfile Types** | The type(s) of DL/I subfile to be processed: NSB and/or NDB. |

# 31 Object Specification - DDMs


- DDMs ..... 180
- DDM Properties ..... 181
- DDM Exceptions ..... 182

This section describes the options provided on the object-specification screens for processing Natural DDMs (data definition modules).

For descriptions of keywords and valid input values, see also `select-clause` in the section *Direct Commands*.

## DDMs

The screen **Unload/Load/Scan DDMs** provides the following fields and PF keys:

| Field | PF Key | Explanation |
|---|---|---|
| **FDIC DBID/FNR** | | Only applies to the unload function.<br><br>The database ID (DBID) and file number of the Adabas file where the DDMs are stored.<br><br>If no values (or 0) are specified, the current FDIC system file is used. |
| **FDIC Password/Cipher** | | Only applies to the unload function.<br><br>The password and cipher code for the Adabas file where the DDMs are stored. |
| **DDM name** | | Only applies to the unload function.<br><br>The name of a DDM or a range of names: see ***Name*** in *Name, Date and Time Specification*. |
| **Select objects** | | Only applies to the unload function.<br><br>Displays a selection list of DDMs available (see also ***Select Objects***). |
| **Properties** | PF7 | Invokes an extra screen where you can specify additional properties of DDMs: see ***DDM Properties***. |
| **Exceptions** | PF8 | Invokes an extra screen where you can specify exceptions to the selection of DDMs: see ***DDM Exceptions***. |
| **Settings** | PF7 | Invokes the **Unload/Load/Scan Settings** screen where you can specify option and parameter settings: see ***Settings***. |

## DDM Properties

The screen **Unload/Load/Scan DDMs, Properties** is used to specify properties for the DDMs selected for processing.

For descriptions of keywords and valid input values, see also `select-clause` in the section *Direct Commands*.

The screen **Unload/Load/Scan DDMs, Properties** provides the following fields:

| Field | Explanation |
|---|---|
| **User ID** | The ID of the user who saved or cataloged a DDM.<br><br>Specify a single user ID or a range of user IDs: see ***Name*** in *Name, Date and Time Specification*. |
| **DDM DBID** | The database ID (DBID) of the DDMs.<br><br>Valid entries are: 1 to 65535 or 0 (all DBIDs) |
| **DDM FNR** | The file number (FNR) of the DDMs:<br><br>Valid entries are: 1 to 65535 or 0 (all FNRs). |
| Object Date:<br><br>**Select all objects (no date check)** | Selects all DDMs, regardless of their date. |
| Object Date:<br><br>**Select objects modified between/and** | See **Object Date** in *Natural Library Object Properties*. |
| Object Date:<br><br>**Select objects modified on** | See **Object Date** in *Natural Library Object Properties*. |
| Object Size:<br><br>**Select all objects (no size check)** | Selects all DDMs, regardless of their size. |
| Object Size:<br><br>**Select objects with size between/and** | Selects all DDMs with a size within the range specified in these fields by entering a start size and/or an end size. |
| Object Size:<br><br>**Select objects with size** | Selects all DDMs with a size that fits the size specified in this field. |

## DDM Exceptions

The screen **Unload/Load/Scan DDMs, Exceptions** is used to specify exceptions to the selection of DDMs.

All objects that match the selection criteria specified in ***DDMs*** and ***DDM Properties*** are checked against the specifications made on the screen **Unload/Load/Scan DDM, Exceptions**. Objects that match *all* specifications defined as exceptions, are exempted from processing.

For descriptions of keywords and valid input values, see also `select-clause` in the section *Direct Commands*.

The screen **Unload/Load/Scan DDMs, Exceptions** provides the following fields:

| Field | Explanation |
|---|---|
| **DDM name** | The name of a DDM or a range of names: see ***Name*** in *Name, Date and Time Specification*. |
| **DDM DBID** | See **DDM DBID** in *DDM Properties*. |
| **DDM FNR** | See **DDM FNR** in *DDM Properties*. |
| **User ID** | See **User ID** in *DDM Properties*. |
| Object Date:<br><br>**Ignore object date** | Performs no date check. DDMs are processed, regardless of their date. |
| Object Date:<br><br>**Exclude objects modified between/and** | See **Object Date** in the section *Natural Library Object Exception Properties*. |
| Object Date:<br><br>**Exclude objects modified on** | See **Object Date** in the section *Natural Library Object Exception Properties*. |
| Object Size:<br><br>**Ignore object size** | Performs no size check. DDMs are processed, regardless of their size. |
| Object Size:<br><br>**Exclude objects with size between/and** | Exempts from processing all DDMs with a size within the range specified in these fields by entering a start size and/or an end size. |
| Object Size:<br><br>**Exclude objects with size** | Exempts from processing all DDMs with a size that fits the size specified in this field. |

# 32 Object Specification - FDTs

This section describes the options provided on the object-specification screen for processing Adabas FDTs (Field Definition Tables).

For descriptions of keywords and valid input values, see also `select-clause` in the section *Direct Commands*.

The screen **Unload/Load/Scan FDTs** provides the following fields and PF keys:

| Field | PF Key | Explanation |
|---|---|---|
| **DBID** | | The database ID where the FDT is located.<br><br>Load and scan:<br>A valid DBID or `0` for all DBIDs. |
| **FNR** | | The file number where the FDT is located.<br><br>Load and scan:<br>A valid FNR or `0` for all FDTs. |
| **Password/Cipher** | | Only applies to the unload and load functions.<br><br>The Adabas password and the cipher code of the Adabas file where the FDT is located. |
| **Settings** | PF7 | Invokes the **Unload/Load/Scan Settings** screen where you can specify option and parameter settings. See ***Settings***. |

# 33 Use Selection or List Workplan

This option is used to specify a Workplan of the type SELECTION or LIST. These Workplans specify selection criteria for the objects to be processed. See also the section ***Workplans***.

The screen **Unload/Load/Scan Selection or List** provides the following fields and PF keys:

| Field | PF Key | Explanation |
|---|---|---|
| **Name** | PF4 | The name of the Workplan to be processed. |
| **List Workplan** | | Displays the contents of the Workplan specified in the **Name** field. |
| **Select Workplan** | PF5 | Displays a list of all Workplans available. See also ***List the Available Workplans in the Workplan Library*** in *Administration*. |
| **Settings** | PF7 | Invokes the **Unload/Load/Scan Settings** screen where you can specify option and parameter settings. See ***Settings***. |

# 34 Settings


- Settings Screen Fields ..... 188
- Set Additional Options ..... 190
- Set Global Parameters ..... 198

The settings option is used to specify option settings for the unload, load, find or scan function and parameter settings for the unload or load function.

### To invoke the Unload/Load/Scan Settings screen

- On any of the unload, load or scan screens, enter the following internal command:

  ```
  SETTINGS
  ```

  See also ***Commands for Navigation and Special Functions*** in the section *Direct Commands*.

  Or:

  Activate advanced-user mode, choose a function and choose ENTER to start the processing procedure.

  Or:

  On advanced-user screens, choose PF7 (Setti).

Unless selected by default, to activate the options provided on the **Unload/Load/Scan Settings** screen described below, mark the corresponding input field with any single character.

## Settings Screen Fields

The **Unload/Load/Scan Settings** screen provides the following fields and PF keys:

| Field | PF Key | Explanation |
|---|---|---|
| **Transfer format** | | Only valid if **Use default options** (this is the default) has been selected.<br><br>If selected, the data to be processed is written/read in Transfer format to/from the work file. See also ***Work File Format*** in *Work Files*.<br><br>Unload function:<br>The data to be unloaded is written in Transfer format to the work file. Note that if you want to change the setting of this field for a subsequent unload, you need to return to the **Main Menu** or enter the command `GO UNLOAD END` (see ***Commands for Navigation and Special Functions*** in *Direct Commands*) and restart the unload function.<br><br>Load and scan functions:<br>The data to be loaded or scanned are expected to be in Transfer format. |

| Field | PF Key | Explanation |
|---|---|---|
| **Unicode work file** | | Only applies to the unload function and if **Transfer format** has been selected.<br><br>If this option is selected, all object sources are converted to Unicode/UTF-8 (Universal Transformation Format, 8-bit form) before they are written to the work file.<br><br>If a Unicode work file is specified, you cannot use the transfer options **Use conversion table**, **Substitute line references** and **Incorporate free rules**. |
| **Use PC File** | | Only applies if Entire Connection is installed.<br><br>If selected, the data to be processed is read from or written to an Entire Connection work file. |
| **PC File** | | Only applies if Entire Connection is installed.<br><br>The complete path name to the Entire Connection work file. If your system environment does not accept a backslash (\) separator, use a slash (/) instead. |
| **Use default options** | | Default options are used (this is the default). See also ***Profile Settings*** and ***Set Additional Options***. |
| **Set additional options** | PF4 | Only valid if **Use default options** has been selected.<br><br>Invokes the **Options** screen where you can modify the default settings and enter additional options for the processing sequence. For the options available, see ***Set Additional Options***. |
| **Use Option Workplan** | | A Workplan of the type OPTION is used. See also ***Workplans***. |
| **Name**<br><br>(next to Use Option Workplan) | | Only valid if **Use Option Workplan** has been selected.<br><br>The name of a Workplan of the type OPTION to be used. |
| **List Option Workplan** | PF6 | Only valid if **Use Option Workplan** has been selected.<br><br>Displays the contents of the Workplan specified in the **Name** field next to **Use Option Workplan**. |
| **Select Option Workplan** | PF5 | Only valid if **Use Option Workplan** has been selected.<br><br>Displays a selection list of available Workplans of the type OPTION (see also ***List the Available Workplans in the Workplan Library*** in *Administration*). |
| **Do not use parameters** | | If selected (default setting), no parameters are set. |
| **Use global parameters** | | Global parameters are used. See ***Set Global Parameters***. |

| Field | PF Key | Explanation |
|---|---|---|
| **Set global parameters** | PF7 | Only valid if **Use global parameters** has been selected.<br><br>Invokes the **Parameters** screen. See ***Set Global Parameters*** and `parameter-setting` (*Direct Commands*) for descriptions of keywords and valid input values. |
| **Use Parameter Workplan** | | A Workplan of the type PARAMETER is used. See also ***Workplans***. |
| **Name**<br><br>(next to Use Parameter Workplan) | | Only valid if **Use Parameter Workplan** has been selected.<br><br>The name of a Workplan of the type PARAMETER to be used. |
| **List Parameter Workplan** | PF9 | Only valid if **Use Parameter Workplan** has been selected.<br><br>Displays the contents of the Workplan specified in the **Name** field next to **Use Parameter Workplan**. |
| **Select Parameter Workplan** | | Only valid if **Use Parameter Workplan** has been selected.<br><br>Displays a selection list of available Workplans of the type PARAMETER. See also ***List the Available Workplans in the Workplan Library*** in *Administration*. |

## Set Additional Options

The sections contained in the **Options** screen are described below. Note that not all of the sections may appear on the screen, because they depend on the function used, the settings defined and the products installed.

For descriptions of keywords and valid input values, see also `option-setting` in the section *Direct Commands*.

This section covers the following topics:

- Work File and Report Options
- XREF Options
- XRef Considerations
- Version Check
- Transfer Options
- Replace Options
- Number to Process
- FDIC Settings

- FSEC Settings

### Work File and Report Options

The options provided for work files and reports are described in the following section.

| Field | Explanation |
|---|---|
| **Use PC File** | Only applies if Entire Connection is installed.<br><br>If selected, the data to be processed is read from or written to an Entire Connection work file. |
| **PC File** | Only applies if Entire Connection is installed.<br><br>The complete path name assigned to the Entire Connection work file. If your system environment does not accept a backslash (\) separator, use a slash (/) instead. |
| **Unicode work file** | Only applies to the unload function and if **Transfer format** has been selected.<br><br>If this option is selected, all object sources are converted to Unicode/UTF-8 (Universal Transformation Format, 8-bit form) before they are written to the work file.<br><br>If a Unicode work file is specified, you cannot use the transfer options **Use conversion table**, **Substitute line references** and **Incorporate free rules**. |
| **Write report** | Writes a report of the objects processed to the report text member specified in the **Report text member** field.<br><br>The **Write report** option is selected by default.<br><br>To display the report, enter the internal command `SHOW REPORT FILE` (see ***Commands for Navigation and Special Functions*** in *Direct Commands*). |
| **Start new report** | Only valid if **Write report** has been selected.<br><br>Deletes the contents of the report text member before a new report is written. Otherwise, a new report is appended to the existing one. |
| **Error report only** | Only valid if **Write report** has been selected.<br><br>Write only error messages to the report. This includes messages from Natural Security and messages that have incurred during the execution of a LOAD command, for instance "not replaced". See also `REPORT-OPTION-1` in *Direct Commands*, *option-setting*. |
| **Change report library** | Only valid if **Write report** has been selected.<br><br>Opens a new screen where you can change the report library.<br><br>See also ***Change the Report Library*** (*Administration*) and `REPORT-LIBRARY` (*Direct Commands*, *option-setting*). |

| Field | Explanation |
|---|---|
| **Report text member** | Only valid if **Write report** has been selected.<br><br>The name of the text member stored in the Workplan library to which the report is written. |
| **Select text member** | Displays a list of all text members stored in the Workplan library. From this list, you can select a **Report text member**. |
| **Write restart information** | Only applies to the load function.<br><br>When this option is set, restart information is provided for the **restart load** function.<br><br>For details, see ***Restart Load*** in *Functions*. |
| **Restart text member** | Only applies to the load function and if **Write restart information** has been selected.<br><br>The name of the text member in the Workplan library to which the restart information is written.<br><br>If you do not specify a name, the Object Handler generates a name and assigns it to the text member. |
| **Select text member** | Displays a list of all text members stored in the Workplan library. From this list, you can select a **Restart text member**. |
| **Delete allowed** | Only applies to the load function. Processes delete instructions from work files when loading objects in internal format. |

## XREF Options

XREF options are only available when unloading or loading data in internal format, that is, if the field **Transfer format** has *not* been selected. Predict must be installed to process XRef data.

The XREF options provided and the functions to which they apply are described in the following section.

| Field | Explanation | Function |
|---|---|---|
| **Yes (unload XRef data)**<br><br>or<br><br>**Yes (load XRef data)** | Unloads cataloged objects and their cross-reference data, if any. Loads cataloged objects and their cross-reference data if cross-references exist in the work file. | Unload<br>Load |
| **No (ignore XRef data)** | No XRef data is processed. | Unload<br>Load |
| **Force** | Loads cataloged objects and their cross-reference data only if cross-references exist in the work file and if Predict entries exist for the objects in the FDIC system file. | Load |
| **Doc** | Loads cataloged objects and their cross-reference data (if any) only if Predict entries exist for the objects in the FDIC system file. | Load |
| **Special** | Loads cataloged objects and their cross-reference data (if any). | Load |

## XRef Considerations

All cross-reference (XRef) data stored in the Predict system file can be processed with the Object Handler. The XREF option indicates whether the Object Handler should process XRef data. XRef data is always deleted if the delete or replace function is performed on a cataloged object.

If Predict has not been installed, set the XREF option to `N` and thus no validation of Predict files is performed. If the XREF option is set to `Y` and the FDIC file being used is not a valid Predict file, an error message is returned.

The rules for setting the XREF option are the same as the ones imposed by Natural Security. In a non-security environment there are no restrictions, see the first five cases described below. However, if Natural Security is active, as in the last case, the setting of the XREF option in the Object Handler depends on the value of the XREF option in the utility profiles of Natural Security.

Consider the following settings for XREF:

- XREF set to OFF or No
- XREF set to ON or Yes or Force
- XREF set to Force
- XREF set to Doc
- XREF set to Special
- XREF option with Natural Security

### XREF set to OFF or No

If the XREF option is set to **OFF** or **No**, no XRef data is processed. But in situations where a cataloged object is deleted or replaced, the Object Handler deletes the XRef data. The target Predict system file is determined according to the current settings of the FDIC option. The default is the value assigned to the profile parameter FDIC (see *FDIC - Predict System File* in the *Parameter Reference* documentation) at the start of the Natural session.

### XREF set to ON or Yes or Force

If the XREF option is set to **Yes** or **Force**, the following actions are applied during processing:

- **Unload**
  Unloads cataloged objects and their cross-reference data (if any).
- **Load**
  Loads cataloged objects and their cross-reference data if cross-references exist in the work file.

#### XREF set to Force

Only applies to LOAD.

Loads cataloged objects and their cross-reference data only if cross-references exist in the work file and if Predict entries exist for the objects in the FDIC system file.

If the XREF option is set to **Force**, the Object Handler additionally checks that the cataloged object has a Predict program entry defined on the Predict system target file. If not, processing of the object is terminated.

#### XREF set to Doc

Only applies to LOAD.

If the XREF option is set to **Doc**, the Object Handler checks that the cataloged object has a Predict program entry defined on the Predict system target file. If not, processing of the object is terminated. The cataloged objects that have cross-reference data are processed with their cross-reference data, and the cataloged objects that have none are also processed.

#### XREF set to Special

Only applies to LOAD.

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

#### XREF option with Natural Security

If Natural Security is active, the following applies:

- If the value of the XREF option in the utility profiles of Natural Security is `N` or `S`, you can specify any value (**OFF/No**, **ON/Yes**, **Doc**, **Force** or **Special**) for the XREF option in the Object Handler.
- If the value of the XREF option in the utility profiles of Natural Security is `D`, you can specify only the values **Doc** or **Force** for the XREF option in the Object Handler. If you specify **ON/Yes**, the value **Force** is used internally. If you specify **OFF/No** or **Special**, the value **Doc** is used internally.
- If the value of the XREF option in the utility profiles of Natural Security is `Y`, you can specify only the values **ON/Yes** or **Force** for the XREF option in the Object Handler. If you specify **Doc**, internally the value **Force** is used, if you specify **OFF/No** or **Special**, the value **ON/Yes** is used internally.

- If the value of the XREF option in the utility profiles of Natural Security is `F`, you can specify only the value **Force** for the XREF option in the Object Handler. If you specify any other value, the value **Force** is used internally.

### Version Check

The **Version check** option is only available when loading data in internal format, that is, if the field **Transfer format** has *not* been selected.

If **Version check** is selected, the Natural version under which the objects were cataloged and written to the work file is compared with the current Natural version. Objects cataloged under a Natural version higher than the current one will be rejected.

### Transfer Options

Transfer options are only available when processing data in Transfer format, that is, if the field **Transfer format** has been selected.

The transfer options provided and the functions to which they apply are described in the following section.

| Option | Explanation | Function |
|---|---|---|
| **Substitute line references** | Only applies if source-code line numbers are used for statement references.<br><br>If line numbers are used as references in the source code, the line numbers of referenced lines and the line number references are replaced with labels. The sources are not modified in the database. | Unload |
| **Include line numbers** | If you choose this option, the line numbers will be transferred. (By default, line numbers in Natural objects are not transferred.) | Unload |
| **Incorporate free rules** | If Predict is installed, Predict rules associated with a map are incorporated into the map source. | Unload |
| **Use conversion table** | **Caution:**<br>Use this option only in special cases, such as when performing a non-FTP transfer between environments with different character sets, where no conversion is performed by the transfer tool.<br><br>Unload:<br>Converts data to ASCII format by using the internal Natural conversion table (**System table**) or a conversion table defined by the user (**User table**).<br><br>Load:<br>Converts data to EBCDIC format by using the internal Natural conversion table (**System table**) or a conversion table defined by the user (**User table**). Note that this only applies if the data in the work file is in ASCII format or if a conversion program is specified (see **User table**). | Unload<br>Load |

| Option | Explanation | Function |
|---|---|---|
| **System table** | Only valid if **Use conversion table** has been selected.<br><br>Unload:<br>Converts data to ASCII format by using the internal Natural conversion table.<br><br>Load:<br>Converts data to EBCDIC format by using the internal Natural conversion table. | Unload<br>Load |
| **User table** | Only valid if **Use conversion table** has been selected.<br><br>If the name of a conversion program has been entered in the field, data is converted to EBCDIC or ASCII format by using the conversion program defined. To specify an individual conversion program, the program must be located in the library SYSOBJH or one of its steplibs. See the example subprograms OTNCONAE and OTNCONEA in the library SYSOBJH.<br><br>If no conversion program is specified, by default, the corresponding conversion table in the Natural file *NATCONV.INI* is used for the unload ([ISO8859_1->EBCDIC] ) and the load ([EBCDIC->ISO8859_1]) functions. | Unload<br>Load |
| **Use load code page** | If you choose this option, a window appears where you can enter the name of the code page to be used for the load function.<br><br>If this option is selected, all object sources unloaded into a work file in UTF-8 will be converted with the specified code page when they are loaded into a work file. See also **Unicode work file**.<br><br>If you enter `*CODEPAGE` as the code page name, the value assigned to the system variable `*CODEPAGE` is used (see the *System Variables* documentation).<br><br>If no code page name is specified, the source objects are converted with the code page used when unloading them.<br><br>If **Use load code page** is specified, you cannot use the options **Use conversion table** and **Translate to upper case**. | Load |
| **Translate to upper case** | Translates any source code to be loaded into upper case. | Load |
| **Data area format** | Only applies to data areas.<br><br>Specifies the format in which to unload or load data area sources. Possible input values are:<br><br>`N` — Converts data areas to the new internal data area format.<br><br>`O` — Converts data areas to the old internal data area format. If one or more data area sources cannot be converted to the old internal data area format, the Object Handler issues a corresponding message when unloading is | Unload<br>Load |

| Option | Explanation | | Function |
|---|---|---|---|
| | | complete. In addition, in the **Status** column of the unload report generated by the unload function, a corresponding remark appears next to the names of the data area sources affected. | |
| | * | Does not convert data areas. This is the default. | |
| | | | |
| | For details, see *Data Area Editor* in the *Editors* documentation. | | |
| **Unshape Arabic characters** | This option is used to replace shaped Arabic characters (code page IBM420) with the corresponding unshaped characters in the following source objects: Natural programs, DDMs, user error messages and system error messages. | | Unload Load |

### Replace Options

The replace options described below only apply to the load function:

| | |
|---|---|
| **Do not replace** | Does not replace any objects. This is the default. |
| **Replace all** | Replaces all objects. |
| **Replace obsolete** | Replaces objects with a date older than the date of the objects in the load file. |
| **Replace except newer** | Replaces all objects except those with a date newer than the date of the objects in the load file. |

### Number to Process

Number to process only applies to the load and scan functions.

In the field **Number to process**, enter a value with a maximum of 5 digits. If a value greater than 0 is specified, the load or scan function stops after the specified number of objects has been processed.

**Note:** If a cataloged Natural object is processed directly after the source object of the same name, they are considered one object.

### FDIC Settings

FDIC settings are used to specify the Predict file (FDIC) to be used for processing XRef data (only applies if Predict is installed) or load DDMs:

| | |
|---|---|
| **DBID** | The database ID where the FDIC file is located. |
| **FNR** | The file number where the FDIC file is located. |
| **Password** | Optional.<br>The Adabas password of the Adabas file where the FDIC file is located. |
| **Cipher** | Optional.<br>The cipher code of the Adabas file where the FDIC file is located. |

### FSEC Settings

FSEC settings only apply if Natural Security is installed.

FSEC settings are used to specify the Natural Security data file (FSEC) to be used for security checks:

| | |
|---|---|
| **DBID** | The database ID where the FSEC file is located. |
| **FNR** | The file number where the FSEC file is located. |
| **Password** | Optional.<br>The Adabas password of the Adabas file where the FSEC file is located. |
| **Cipher** | Optional.<br>The cipher code of the Adabas file where the FSEC file is located. |

## Set Global Parameters

**Only applies to the load or unload function.**

The fields provided on the **Parameters** screen can be used to change global parameter settings for the objects to be processed with the load or unload function, and to change the target environment for the load function. For example, you can specify new names (or name ranges) under which the selected objects are unloaded to the work file, or you can specify a different library into which the selected objects are loaded from the work file.

If global parameters are specified during the unload function, the parameter settings affect the objects before they are written to the work file. If they are specified during the load function, the parameter settings affect the objects before they are written to the target environment.

The values that can be specified to change parameter settings, are entered next to the required parameters in the fields **Check Value** and **New Value**.

If no value has been entered in **Check Value**, the value entered in **New Value** affects all objects to which the specific parameter setting applies. If a value has been entered in **Check Value**, the value entered in **New Value** only affects objects to which the specific parameter setting and the value entered in **Check Value** apply. If a **Check Value** or **New Value** is not relevant to the type of object to be processed, any value entered in either field will be ignored. For example: Natural

system error messages have no library name. Therefore, when processing Natural system error messages, a value entered in **Check Value** or **New Value** for the **Library** field will be ignored.

**Check Value** and **New Value** do not apply to the parameter **Error number difference** and the parameters contained in the section **System files for load** of the **Parameters** screen.

For valid parameter settings, see also `parameter-setting` in the section *Direct Commands*.

The following fields are contained in the **Parameters** screen:

| Field/Section | Explanation |
|---|---|
| **Object name** | **Check Value/New Value:**<br><br>A single object name or a range of names: see ***Name*** in *Name, Date and Time Specification* and ***Rules for New Values***.<br><br>**Note:** Not applicable to DDMs on mainframe platforms. |
| **Library** | **Check Value/New Value:**<br><br>A single library name or a range of names: see ***Name*** and ***Rules for New Values***. |
| **Date** | **Check Value/New Value:**<br><br>A single date or a range of dates: see ***Date*** in *Name, Date and Time Specification* and ***Rules for New Values***. |
| **Time** | **Check Value/New Value:**<br><br>A time or a range of times: see ***Time*** in *Name, Date and Time Specification* and ***Rules for New Values***. |
| **User ID** | **Check Value/New Value:**<br><br>A single user ID or a range of user IDs: see ***Name*** and ***Rules for New Values***. |
| **Terminal ID** | **Check Value/New Value:**<br><br>A single terminal ID or a range of terminal IDs: see ***Name*** and ***Rules for New Values***. |
| **Lang. codes** | Only applies when processing Natural system error messages or user-defined error messages.<br><br>**Check Value/New Value:**<br><br>Up to 8 valid language codes such as code `4` for Spanish. If more than one language code is specified, **Check Value** must contain the same number of language codes. In this case, the language code in **Check Value** is replaced by the language code in the corresponding **New Value**.<br><br>Note: **New Value** does not apply to the long texts of Natural system error messages for which English (code `1`) is the only valid language. |

| Field/Section | Explanation |
|---|---|
| **Error number difference** | Only applies when processing Natural system error messages or user-defined error messages.<br><br>A 4-digit positive or negative value (+/-*nnnn*) to be used as a new number range for error messages. Start and end values must be provided in the **Error number from/to** fields (see *Natural Library Objects*) to validate whether the new range can be applied to the selected error messages.<br><br>Example:<br><br>If **Error number from/to** selects message numbers 1 to 10 and **Error number difference** is set to `2000`, the messages will be renumbered from 2001 to 2010. A value of `-1000` in **Error number difference** would cause a validation error. |
| **FDT DBID/FNR** | **Check Value/New Value:**<br><br>A valid database ID (DBID) and/or file number (FNR) for Adabas FDTs. |
| System files for load: Load FNAT<br><br>**DBID**<br>**FNR**<br>**Password**<br>**Cipher** | Only applies to the load function.<br><br>The database ID (**DBID**) and file number (**FNR**) of the target FNAT system file. This system file is used for all library objects whose library name starts with `SYS`, but not SYSTEM. If required, enter the Adabas password (**Password**) and the cipher code (**Cipher**) for the system file. |
| System files for load: Load FUSER<br><br>**DBID**<br>**FNR**<br>**Password**<br>**Cipher** | Only applies to the load function.<br><br>The database ID (**DBID**) and file number (**FNR**) of the target FUSER system file. This system file is used for all library objects whose library name does not start with `SYS`, and for the library SYSTEM. If required, enter the Adabas password (**Password**) and the cipher code (**Cipher**) for the system file. |
| System files for load: Load FNAT/FUSER<br><br>**Select** | Only applies to the load function.<br><br>Invokes the **Select System File** window with a list of all system files available in your Natural environment: see ***Select System File***. |
| System files for load: Load NCP<br><br>**DBID**<br>**FNR**<br>**Password**<br>**Cipher** | Only applies to the load function.<br><br>The database ID (**DBID**) and file number (**FNR**) of the target Adabas file into which the Natural command processor sources are to be loaded. If required, enter the Adabas password (**Password**) and the cipher code (**Cipher**) for the system file. |

This section covers the following topic:

- Rules for New Values

## Rules for New Values

The following applies to **New Value** for **Object name**, **Library**, **Date/Time**, **User ID** and **Terminal ID**.

If **New Value** contains a range with an asterisk (*) such as `ABC*`, the number of characters before the asterisk (*) determines the number of characters to be replaced in **Check Value**. This is also valid if **Check Value** is shorter than the range specified in **New Value** (see the second example in *Examples* below).

**Examples:**

1. If **Object name** is `ABCDEFG` and **New Value** is set to `ZYX*`, the resulting object name is ZYXDEFG.
2. If **Object name** is `AB` and **New Value** is set to `ZYX*`, the resulting object name is ZYX.
3. If **Date/Time** is `2005-03-26` and **New Value** is set to `2006*`, the resulting object date is 2006-03-26.

# 35 Workplans


- Creating, Selecting and Modifying Workplans ..... 204
- Contents of Workplans ..... 204
- Examples of Workplans ..... 205
- Referencing Workplans ..... 206

Workplans define individual standard procedures for command execution, object selection and parameter or option settings which can be used to further automate function processing.

Workplans are Natural objects of the type Text. They are, by default, stored in the library WORKPLAN located in the current FUSER system file.

## Creating, Selecting and Modifying Workplans

You can use the **administration** function (see the relevant section) to create a Workplan, select a Workplan from a list, modify a Workplan, or change the default library for Workplans. The default library can also be changed by specifying the `Workplan-Library` parameter in your Object Handler profile (see ***Profile Settings***).

## Contents of Workplans

A Workplan consists of a header (generated by the Object Handler) and an associated instructional or textual part. Instructional parts contain Object Handler commands and parameter and/or option settings. Textual parts contain plain text only. Header and instructional or textual parts can contain comments (for example, the short description of the Workplan) that must start with the delimiter characters `/*` and are restricted to one line.

There are six types of Workplan: PROCEDURE, SELECTION, LIST, PARAMETER, OPTION and TEXT.

The table below lists the valid headers (to be entered if creating a Workplan outside the Object Handler) for the corresponding types of Workplan and describes the contents of the instructional or textual part. Additionally, it provides cross references to the clauses that apply when specifying Object Handler direct commands. The Object Handler direct commands provided are explained in the section ***Direct Commands***.

| Valid Headers | Contents | Related Topic in *Direct Commands* |
|---|---|---|
| `TYPE PROCEDURE` | An Object Handler command procedure.<br><br>This Workplan can contain any combination of Object Handler commands available for PROCEDURE. Enter a sequence of commands separated by semicolons (;). | ***Basic Command Syntax*** |
| `TYPE SELECTION` | Selection criteria for objects.<br><br>This Workplan can be used in Object Handler Workplan commands. | `select-clause` |

| Valid Headers | Contents | Related Topic in *Direct Commands* |
|---|---|---|
| `TYPE LIST` | A list of objects.<br><br>This Workplan can be used in Object Handler Workplan commands. | *`select-clause`*<br><br>***Object List - LIST Workplan*** |
| `TYPE PARAMETER` | Parameters for the unload or load function.<br><br>This Workplan can be used to change attributes for the objects to be processed such as the name of a new target library where objects are loaded.<br><br>`TYPE PARAMETER` can be used in Object Handler Workplan commands. | *`parameter-setting`* |
| `TYPE OPTION` | Options for the unload or load function, for example, report settings.<br><br>This Workplan can be used in Object Handler Workplan commands. | *`option-setting`* |
| `TYPE TEXT` | Comments or any other text that can be used for documentation purpose. | Not applicable |

## Examples of Workplans

The following table lists examples of instructional parts contained in a Workplan.

| Workplan Type | Instruction | Explanation |
|---|---|---|
| PROCEDURE | `FINDLIB * LIB TEST` | Check whether the library TEST exists. |
| PROCEDURE | `UNLOAD A* LIB TEST` | Unload from the library TEST into Work File 1 all Natural programming objects starting with `A`, and all user-defined error messages. |
| SELECTION | `* LIB TEST` | Process all objects from the library TEST. |
| TEXT | `This is a Workplan comment.` | Any text. |

This section covers the following topic:

- Example of Workplan Contents

### Example of Workplan Contents

The following is an example listing of a PROCEDURE Workplan where the `UNLOAD` command is executed:

```
TYPE PROCEDURE /* VERSION=03.01 NATURAL VERSION=08.02.01 PL=0 AUTHOR=SAG ↩
DATE=2010-07-20 09:40:12
/* unload from library TEST with target library PROD01
UNLOAD * LIB TEST OBJTYPE N
WITH NEWLIBRARY PROD01
WHERE REPORT MYREP01
```

## Referencing Workplans

You can reference a Workplan by using Object Handler menu functions or direct commands (see also the section ***Direct Commands***).

The following syntax applies when referencing a Workplan with the Object Handler direct commands described in the section ***Direct Commands***.

```
( workplan-name
  [ LIBRARY library-name ]
  [DBID dbid [FNR fnr ]] [NAME vsam-name ]
  [CIPHER cipher ]
  [ { PASSWORD } password ]
  [ { PSW }                ]
)
```

The syntactical options are explained in the following section:

- Keyword Explanation

## Keyword Explanation

The table below describes the keywords and values that apply to the syntax for referencing Workplans.

| Keyword | Values | Default Value |
|---|---|---|
| *workplan-name* | The name of the Natural text member in the Workplan library to be used as the Workplan. | No default |
| LIBRARY | The name of the library where the Workplan is located. | WORKPLAN |
| DBID | The ID of the Adabas database where the Workplan library is located. | 0<br>(current FNAT/FUSER) |
| FNR | The number of the Adabas file where the Workplan library is located. | 0<br>(current FNAT/FUSER) |
| NAME | Only applies to objects on mainframes.<br><br>The name of a valid VSAM file where the Workplan library is located. | blank<br>(current FNAT/FUSER) |
| CIPHER | Only applies to objects on mainframes.<br><br>An 8-digit cipher code. | blank<br>(current FNAT/FUSER) |
| PASSWORD | Only applies to objects on mainframes.<br><br>An 8-character Adabas password. | blank<br>(current FNAT/FUSER) |

# 36 Name, Date and Time Specification


- Name ..... 210
- Date ..... 211
- Time ..... 212

You can use a name, a date, a time or a range of names, dates and times to select Natural library objects, Natural command processor sources, Natural-related objects or Natural DDMs (data definition modules).

## Name

You can specify a name or a range of names.

In the list of options below, *value* is any combination of one or more characters:

| | Input | Items Selected |
|---|---|---|
| | *value* | All items with names equal to *value*. |
| | *<br>> | All items. |
| | ? | All items with any single character for each question mark (?) entered. |
| Leading Characters | *value** | All items with names that start with *value*.<br><br>Example: AB*<br>Selected: AB, AB1, ABC, ABEZ<br>Not selected: AA1, ACB |
| Wildcard | *value*? | All items with names that start with *value* and end with any single character for each question mark (?) entered.<br><br>Example: ABC?<br>Selected: ABCA, ABCZ<br>Not selected: AXC, ABCAA |
| | *value*?*value*? | All items that match *value* combined with asterisk (*) and question mark (?) in any order.<br><br>Example: A?C*Z<br>Selected: ABCZ, AXCBBBZ, ANCZ<br>Not selected: ACBZ, ABDEZ, AXCBBBZA |
| | *value**value*? | |
| | **value*?*value** | |
| Start Value | *value*> | All items with names greater than or equal to *value*.<br><br>Example: AB><br>Selected: AB, AB1, BBB, ZZZZZZZ<br>Not selected: AA1, AAB |
| End Value | *value*< | All items with names less than or equal to *value*.<br><br>Example: AX<<br>Selected: AB, AWW, AX<br>Not selected: AXA, AY |

**Note:** The parameter specification option **New Value** only allows leading characters (asterisk (*) notation). See ***Rules for New Values*** in *Set Global Parameters* in the section *Settings*.

## Date

All date values within the Object Handler are specified in international date format.

You can specify a date, a range of dates, a special date or a range of special dates. A date must be specified in the format *YYYY-MM-DD* (*YYYY* = year, *MM* = month, *DD* = day).

In the list of options below, the underlined portion of a keyword represents its valid abbreviation, and *value* is any combination of one or more digits:

| | Input Value | Items Selected |
|---|---|---|
| Date | *YYYY-MM-DD* | All items with a date equal to *YYYY-MM-DD*.<br><br>Example: 2003-02-15 |
| Leading characters | *value** | All items with a date that starts with *value*.<br><br>Example: 2002*<br>Selected: 2002-01-01, 2002-12-31<br>Not selected: 2001-12-31, 2003-01-01 |
| Start value | *value*> | All items with a date greater than *value*.<br><br>Example: 2002-05><br>Selected: 2002-05-01, 2002-12-31, 2003-01-01, 2003-12-31<br>Not selected: 2002-04-31, 2001-12-31<br><br>**Special dates** can be used as *value* (see below). |
| End value | *value*< | All items with a date less than *value*.<br><br>Example: 2003-02<<br>Selected: 2002-05-01, 2002-12-31, 2003-01-01, 2003-01-31<br>Not selected: 2003-02-01, 2003-05-18<br>**Special dates** can be used as *value* (see below). |
| **Special Dates** | | |
| TODAY<br>(+/-*nnnn*) | | All items with the date of the current day.<br><br>The day can be followed by +*nnnn* or -*nnnn* where *nnnn* has a maximum of 4 digits.<br><br>The resulting date is computed as the date of the current day plus or minus *nnnn* days.<br><br>Example: If the current date is 2003-03-01, TODAY +5 results in 2003-03-06. |

| | Input Value | Items Selected |
|---|---|---|
| YESTERDAY | | All items with the date of the day before the current day. |
| MONTH | | All items with the date range of the current month.<br><br>Example: The current month is 2003-02.<br>Selected: 2003-02-01, 2003-02-30<br>Not selected: 2003-03-01 |
| | | FMDATE: Starts with the first day of the current month.<br><br>TODATE: Ends with the last day of the current month.<br><br>If the values of FMDATE and TODATE are identical, the selection is restricted to one day. |
| YEAR | | All items with the date range of the current year.<br><br>Example: The current year is 2003.<br>Selected: 2003-01-01, 2002-12-31<br>Not selected: 2002-31-12 |
| | | FMDATE: Starts with the first day of the current year.<br><br>TODATE: Ends with the last day of the current year.<br><br>If the values of FMDATE and TODATE are identical, the selection is restricted to one year. |

**Note:** The parameter specification option **New Value** only allows leading characters (asterisk (*) notation). See ***Rules for New Values*** in *Set Global Parameters* in the section *Settings*.

## Time

You can specify a time or a range of times. The time must be specified in the format *HH*:*II*:*SS* (*HH* = hours, *II* = minutes, *SS* = seconds).

In the list of options below, *value* is any combination of one or more digits:

| | Input Value | Items Selected |
|---|---|---|
| Time | *HH*:*II*:*SS* | All items with a time equal to *HH*:*II*:*SS*.<br><br>Example: 14:15:16 |
| Leading characters | *value** | All items with a time that starts with *value*.<br><br>Example: 13:*<br>Selected: 13:00:00, 13:10:53, 13:59:59<br>Not selected: 12:59:59, 14:00:00 |

# 37 Work Files


- Work File Assignment ..... 214
- Work File Format ..... 214

This section describes work files and valid formats that apply to the unload, load and scan functions of the Object Handler.

See also ***Work File Options*** in the section *Settings*.

## Work File Assignment

The following table lists the work files used by the Object Handler.

| File | Explanation |
|---|---|
| Work File 1 | Used for the unload, load and scan functions.<br><br>Contains the data unloaded. |
| Work File 7 | Only used if Entire Connection is installed and if **Use PC File** is selected on the **Options** screen (see also ***Set Additional Options***).<br><br>Work File 7 must be defined as Entire Connection work file to be used for the unload, load and scan functions.<br><br>Contains the data unloaded. |
| Work File 8 | Only used if Entire Connection is installed and if **Use PC File** is selected on the **Options** screen (see also ***Set Additional Options***).<br><br>Work File 8 must be defined as Entire Connection work file to be used for the unload, load and scan functions.<br><br>Used as internal file for processing Entire Connection commands.<br><br>**Note:**<br><br>The number of the work file can be changed with user exit routine **OBJHEX03** (see *Batch Condition Codes and User Exit Routines*) or with the options `PCCOMMANDFILENUMBER`, `PCCOM` and `PCCFN` (see *`option-setting`*) when using a Workplan of the type PROCEDURE. |

## Work File Format

There are two file formats for unloading objects in the source environment into work files and for loading them from work files into the target environment: an internal format and the Transfer format. Work files must be of internal format to transfer binary data. Work files must be of Transfer format to transfer text data.

This section covers the following topics:

- Internal Format

- Transfer Format

### Internal Format

The internal format is an internal record layout for work files that are used to transfer Natural sources and cataloged objects, error messages, Natural command processor sources, Adabas FDTs (Field Definition Tables), Natural DDMs (data definition modules) and Natural-related objects from one environment to another.

Use work files of internal format to transfer objects between identical platforms.

With the internal format activated, Natural objects are read from the source environment and written to a Natural work file by using the unload function of the Object Handler. This work file can be transported to another environment with standard file transfer services. In the target environment, the objects can then be read from the work file and loaded into the local file or database system with the load function of the Object Handler.

We strongly recommend that you define Work File 1 with `RECFM=VB` (variable blocked) because the Object Handler writes records of variable length to the work file, which reduces the work file size.

If you define Work File 1 with `RECFM=FB` (fixed blocked), you must specify the option `FIXEDLENGTH` for the unload function. Otherwise, the records written to the work file are filled up with binary zeros.

**Note:** Work files created by the utility NATUNLD on the server, must be processed in internal format. The work files must be created on a server of the same platform where NATUNLD was applied.

### Transfer Format

See also **Transfer format** in the section *Settings*.

The Transfer format is a general record layout for work files that contain load or unload data. This format is platform-independent and can be used to transfer the sources of Natural objects, Natural command processor sources, error messages, DDMs and Adabas FDTs from one hardware platform to another and between UNIX, OpenVMS, mainframe and Windows platforms.

With the option **Transfer format** set, the unload function of the Object Handler reads Natural objects from a hardware platform and then restructures them.

Formatted records are written to a Natural work file that can be transported to another platform with standard file transfer services. On the target platform, the load function of the Object Handler then reads the objects from the work file and loads them into the local file or database system. The objects read from the work file are restructured according to the structure of the new hardware platform.

### Handling Sources in Unicode/UTF-8

Transfer format is also used to unload or load sources of Natural objects in Unicode/UTF-8 (Universal Transformation Format, 8-bit form). If you specify the corresponding unload option (`WORKFILETYPE` set to `UTF-8` in command mode or **Unicode work file** in menu mode), all object sources will be unloaded into a work file in UTF-8. If you specify the corresponding load option (`LOAD-CODE-PAGE` in command mode or **Use load code page** in menu mode), all object sources in UTF-8 will be converted with the specified code page when they are loaded into a Natural system file.

### Work Files from SYSTRANS

Use Transfer format to process work files created by the utility SYSTRANS. Work files that contain object sources encoded in UTF-8 cannot be processed with SYSTRANS.

# 38 Direct Commands

The Object Handler provides direct commands for the following purposes:

- To execute an Object Handler function such as unloading or loading objects in batch mode or in direct command online mode without using Object Handler menus (see also ***Batch or Direct Command Calls***).
- To execute or reference a Workplan (see also the section ***Workplans***).
- To be used as an instruction in a Workplan.
- To navigate through screens.
- To perform special functions.

This section describes the basic command syntax and the individual clauses, parameter and option settings available to perform these tasks. In addition, you can view examples that illustrate the use of direct commands.

The symbols used in the syntax diagrams shown in this section are explained in *System Command Syntax* in the *System Commands* documentation.

This section covers the following topics:

**Basic Command Syntax**

`select-clause`

**Object List - LIST Workplan**

`parameter-setting`

`option-setting`

**Examples of Using Direct Commands**

**Commands for Navigation and Special Functions**

# 39 Basic Command Syntax

This section describes the Object Handler direct commands provided for executing Object Handler functions and Workplans of the type PROCEDURE. It also describes the commands used for migrating from the old utilities NATUNLD/NATLOAD and SYSTRANS to the Object Handler.

For explanations of the variable values contained in the syntax diagrams shown in this section, refer to the relevant sections in the *Object Handler* documentation. For explanations of the symbols used in the syntax diagrams, see *System Command Syntax* in the *System Commands* documentation.

```
EXECUTE (procedure-workplan)
```

Executes a Workplan of the type PROCEDURE. See also the section ***Workplans***.

```
UNLOAD select-clause [parameter-setting] [option-setting]
```

Unloads the objects defined in the `select-clause` with the parameters defined in `parameter-setting` with the options defined in `option-setting`.

```
LOAD select-clause [parameter-setting] [option-setting]
```

Loads the objects defined in the `select-clause` with the parameters defined in `parameter-setting` with the options defined in `option-setting`.

```
LOADALL [parameter-setting] [option-setting]
```

Loads all objects from a work file with the parameters defined in `parameter-setting` with the options defined in `option-setting`.

```
SCAN  select-clause  [option-setting]
```

Scans a work file for the objects defined in the *select-clause* with the options defined in *option-setting*.

```
SCANALL  [option-setting]
```

Scans a work file for all objects with the options defined in *option-setting*.

```
FIND  select-clause  [option-setting]
```

Finds the objects defined in the *select-clause* with the options defined in *option-setting* and writes a report of the objects found into a Natural text member stored in the Workplan library. In addition, a report of the objects found can be written to a specified report file as Natural text member in the Workplan library.

```
FINDLIB  select-clause  [option-setting]
```

Finds the libraries for Natural objects or Natural command processor sources defined in the *select-clause* with the options defined in *option-setting* and writes a report of the objects found into a Natural text member stored in the Workplan library. In addition, a report of the objects found can be written to a specified report file as Natural text member in the Workplan library.

```
DELETE  select-clause  [option-setting]
```

Deletes the objects defined in the *select-clause* with the options defined in *option-setting*.
**Restriction:** It is not possible to delete an FDT.

```
UNDELI  select-clause [option-setting]
```

Unloads delete instructions for the objects defined in the *select-clause* with the options defined in *option-setting*.

```
RESTART  [restart-text-member]
```

Continues an interrupted load function. This is only possible if information was written to a Natural text member in the Workplan library during the aborted load. See also RESTART in the section *option-setting* (*Direct Commands*) and ***Restart Load***.

`DISPLAY STATISTICS`

Displays statistics information about the objects processed.

`NATUNLD natunld-direct-command`

Executes an Object Handler command in the syntax of the old utility NATUNLD. See also ***Migration from NATUNLD/NATLOAD and SYSTRANS to the Object Handler***.

`NATLOAD natload-direct-command`

Executes an Object Handler direct command issued in the syntax of the old utility NATLOAD. See also ***Migration from NATUNLD/NATLOAD and SYSTRANS to the Object Handler***.

`SYSTRANS systrans-direct-command`

Executes an Object Handler direct command issued in the syntax of the old utility SYSTRANS. See also ***Migration from NATUNLD/NATLOAD and SYSTRANS to the Object Handler***.

# 40 select-clause


- Syntax of select-clause ..... 224
- SELECTION or LIST Workplan ..... 224
- Natural Library Object and DDM Selection ..... 225
- Natural-Related Debug Environment Selection ..... 231
- Natural-Related Profile Selection ..... 232
- Natural-Related DL/I Subfile Selection ..... 234
- Natural System Error Message Selection ..... 236
- Natural Command Processor Selection ..... 237
- FDT Selection ..... 239
- Application Selection ..... 240
- Object Selection for Delete Instructions ..... 243
- Help Text Selection ..... 245

The `select-clause` comprises either a Workplan of the type SELECTION or LIST, or selection specifications for the objects, FDTs or applications to be processed.

This section describes the syntax that applies to the `select-clause`. The keywords and variable values contained in the syntax diagrams represent the parameters that can be used to specify object selection criteria. If indicated, a variable value must be supplied with a keyword.

## Syntax of select-clause

The `select-clause` consists of one of the following options:

```
{ (selection-workplan)
  (list-workplan)
  object-selection
  delete-instruction-selection
  help-text-selection          }
```

The `selection-workplan` and `list-workplan` options are explained in *SELECTION or LIST Workplan* below.

The use of `object-selection` depends on the object type, DDM, FDT or application you want to process, for each of which the appropriate syntax and keywords are explained in the remainder of this section.

The `delete-instruction-selection` options are explained in ***Delete Instructions for Selected Objects***.

The `help-text-selection` option is explained in ***Help Text Selection***.

## SELECTION or LIST Workplan

A Workplan of the type SELECTION contains a header (`TYPE SELECTION`) and a selection from one of the following types of object or file: Natural library objects, Natural-related objects, Natural system error messages, Natural command processor sources, DDMs or Adabas FDTs (Field Definition Tables).

A Workplan of the type LIST contains a header (`TYPE LIST`) and a selection list of objects as described in the section ***Object List - LIST Workplan***. Such an object list can be used for the `UNLOAD`, `LOAD` or `FIND` command only.

For further information on using Workplans, see the section ***Workplans***.

## Natural Library Object and DDM Selection

This selection is used to select Natural objects for processing including Natural DDMs (data definition modules) and user-defined error messages.

The appropriate syntax is shown and explained in the following section.

- Syntax of Natural Library Object and DDM Selection

### Syntax of Natural Library Object and DDM Selection

```
object-name
LIBRARY library-name
[ DBID dbid FNR fnr                                          ]
[ [NAME vsam-name]   [ { PASSWORD } password ] ]
[ [CIPHER cipher]                                            ]
[OBJTYPE group-type]
[ SETNO set-number [SETUSER set-user] [SETLIBRARY set-library] ]
[NATTYPE object-type]
[SCKIND object-kind]
[MODE object-mode]
[FMNUM error-number-from]
[TONUM error-number-to]
[SLKIND message-type]
[LANGUAGE languages]
[DDMDBID ddm-dbid] [DDMFNR ddm-fnr]
[NATVERS natural-version]
[ DATE date                                  ]
[ [FMDATE date-from] [TODATE date-to]        ]
[ [SIZE size]                                ]
[ [FMSIZE size-from] [TOSIZE size-to]        ]
[USERID user-id]
[TID terminal-id]
[except-clause]
```

*except-clause*

```
EXCEPT
( object-name
[LIBRARY library-name]
[OBJTYPE group-type]
[SCKIND object-kind]
[NATTYPE object-type]
[MODE object-mode]
[SLKIND message-type]
[FMNUM error-number-from] [TONUM error-number-to]
[LANGUAGE languages]
[DDMDBID ddm-dbid] [DDMFNR ddm-fnr]
[NATVERS natural-version]
[                    DATE date                                  ]
[                    [FMDATE date-from] [TODATE date-to]        ]
[                    SIZE size                                  ]
[                    [FMSIZE size-from] [TOSIZE size-to]        ]
[USERID user-id]
[TID terminal-id]
)
```

**Notes:**

1. For the command `FINDLIB`, only the following keywords are processed: `LIBRARY`, `DBID`, `FNR`, `NAME`, `CIPHER` and `PASSWORD` or `PSW`.
2. When processing Natural DDMs, you must set `OBJTYPE` to `D`. In addition, some keywords do not apply to DDMs, which is described below.

### Keyword Explanation of Natural Library Object and DDM Selection

The keywords and valid values for the objects to be processed are described in the following section.

| Keyword | Valid Values | Default Value |
|---|---|---|
| *object-name* | A valid object name or a range of names.<br>If *object-name* contains blank characters, it must be enclosed in double quotation marks (" ").<br><br>See also ***Name*** in *Name, Date and Time Specification*. | none |

| Keyword | Valid Values | Default Value |
|---|---|---|
| LIBRARY | A valid library name or a range of names.<br><br>If OBJTYPE (see below) is set to D, the library name is ignored.<br><br>If SETNO is specified, a range of names is not allowed.<br><br>See also ***Name***. | none |
| DBID | Not valid for DDMs on mainframes (OBJTYPE set to D; see below).<br><br>A valid database ID. | 0<br>(current FNAT/FUSER) |
| FNR | Not valid for DDMs on mainframes (OBJTYPE set to D; see below).<br><br>A valid file number. | 0<br>(current FNAT/FUSER) |
| NAME | Only applies to objects on mainframes. Not valid for DDMs on mainframes (OBJTYPE set to D; see below).<br><br>A valid VSAM name. | blank<br>(current FNAT/FUSER) |
| CIPHER | Only applies to objects on mainframes. Not valid for DDMs on mainframes (OBJTYPE set to D; see below).<br><br>The 8-digit cipher code of the Adabas file where the objects are stored. | blank<br>(current FNAT/FUSER) |
| PASSWORD<br><br>or<br>PSW | Only applies to objects on mainframes.<br><br>Not valid for DDMs on mainframes (OBJTYPE set to D; see below).<br><br>An 8-character Adabas password. | blank<br>(current FNAT/FUSER) |
| OBJTYPE | Types of object are:<br><br>D DDMs<br>E User-defined error messages<br>N Natural programming objects<br>* Asterisk (all)<br><br>or a valid combination.<br><br>**Exception:**<br>Object type D cannot be combined with any other type. | * |
| SETNO | Only applies to the unload and find functions and if Predict is installed. Not applicable to application objects (see also ***Selecting Application Objects***).<br><br>A one- or two-digit number that identifies the retained set to be used for the names of the objects to be processed. A retained set is created with the save set option of the LIST XREF command.<br><br>If SETNO is specified, the value specified for *object-name* is ignored. | none |

| Keyword | Valid Values | Default Value |
|---|---|---|
| | For detailed information on Predict sets, refer to the *Predict* documentation. | |
| SETUSER | Only applies to the unload and find functions and if Predict is installed. Not applicable to application objects (see also ***Selecting Application Objects***).<br><br>The ID of the user who created the Predict set. If no ID is specified, the value of the system variable *USER (see also the *System Variables* documentation) is used. | *USER |
| SETLIBRARY | Only applies to the unload and find functions and if Predict is installed. Not applicable to application objects (see also ***Selecting Application Objects***).<br><br>The name of the library to be searched for a Predict set. If you do not specify SETLIBRARY, the library specified with LIBRARY is used instead. | |
| NATTYPE | Not applicable if OBJTYPE is set to D.<br><br>One or more single-character codes for Natural object types:<br><br>P Program<br>N Subprogram<br>S Subroutine<br>C Copycode<br>H Helproutine<br>T Text<br>7 Function<br>8 Adapter<br>G Global data area<br>L Local data area<br>A Parameter data area<br>M Map<br>4 Class<br>3 Dialog<br>5 Natural command processor<br>9 Resource<br>* All object types | * |
| SCKIND | Not applicable if OBJTYPE is set to D.<br><br>The kind of Natural programming objects.<br>Valid input values are:<br><br>S Source objects: objects that are only stored in source form. | A |

| Keyword | Valid Values | Default Value |
|---|---|---|
| | C Cataloged objects: objects that are only stored in cataloged form.<br>A All source and cataloged objects.<br>W All STOWed objects: source and cataloged objects with identical date and time.<br>B Source and cataloged objects if both exist.<br><br>**Note:** W and B are valid for the UNLOAD and FIND commands only. For LOAD and SCAN, W and B are valid entries, but they are treated like A (all objects). If data is processed in Transfer format, only S (source objects) or A applies. | |
| MODE | Not applicable if OBJTYPE is set to D.<br><br>The programming mode of the Natural programming objects.<br>Valid input values are:<br><br>A Any.<br>R All objects in reporting mode.<br>S All objects in structured mode. | |
| FMNUM | A start number of Natural error messages.<br><br>Valid range: 1 to 9999. | 1 |
| TONUM | An end number of Natural error messages.<br><br>Valid range: 1 to 9999.<br><br>The value must be greater than or equal to the value of FMNUM, if specified. | 9999<br>or value of FMNUM<br>(if specified) |
| SLKIND | The kind of Natural error message text.<br>Valid input values are:<br><br>S Short text.<br>Cannot be applied to the DELETE command (see ***Basic Command Syntax***).<br>L Long text.<br>A Short and/or long text.<br>B Short and long text, if both exist. | A |
| LANGUAGE | Up to 8 valid language codes (for example, code 1 for English) of user-defined error messages. An asterisk (*) selects all language codes. | * |
| DDMDBID | The valid database ID (1 to 65535) of a DDM.<br><br>UNLOAD, LOAD and SCAN: 0 denotes that no check is performed. DDMs are processed, regardless of their database ID (DBID). | 0 |

| Keyword | Valid Values | Default Value |
|---|---|---|
| DDMFNR | The valid file number (1 to 65535) of a DDM.<br><br>UNLOAD, LOAD and SCAN: 0 denotes that no check is performed. DDMs are processed, regardless of their file number (FNR). | 0 |
| NATVERS | The Natural version of Natural programming objects. You can also specify a range of versions: see ***Name***. | blank<br>(no check) |
| DATE | The save or catalog date of Natural programming objects.<br><br>You can add a time by inserting a blank between date and time. For the format and ranges allowed, see ***Date*** and ***Time*** in *Name, Date and Time Specification*.<br><br>Special terms allowed are YESTERDAY and TODAY. See ***Special Dates*** in *Date*. | blank<br>(no check) |
| FMDATE | A start value:<br><br>The date on or after which Natural programming objects were cataloged or saved. The format is identical to DATE. See ***Date***.<br><br>Special terms allowed are YEAR, MONTH, YESTERDAY and TODAY. See ***Special Dates*** in *Date*. | blank<br>(no check) |
| TODATE | An end value:<br><br>The date on or before which Natural programming objects were cataloged or saved. The format is identical to DATE. See ***Date***.<br><br>Special terms allowed are YEAR, MONTH, YESTERDAY and TODAY. See ***Special Dates*** in *Date*. | blank<br>(no check)<br>or high value<br>(if FMDATE specified) |
| SIZE | The size of Natural programming objects (up to 7 digits). | 0<br>(no check) |
| FMSIZE | A start value:<br><br>The minimum size of Natural programming objects (up to 7 digits). | 0<br>(no check) |
| TOSIZE | An end value:<br><br>The maximum size of Natural programming objects (up to 7 digits). | 0<br>(no check)<br>or high value<br>(if FMSIZE specified) |
| USERID | The ID of the user who saved or cataloged the Natural programming objects.<br><br>You can also specify a range of user IDs: see ***Name***. | blank<br>(no check) |
| TID | Not applicable if OBJTYPE is set to D.<br><br>The ID of the terminal where the Natural programming objects were saved or cataloged (provided by the Natural system variable *INIT-ID). | blank<br>(no check) |

| Keyword | Valid Values | Default Value |
|---|---|---|
| | You can also specify a range of terminal IDs: see also ***Name***. | |
| EXCEPT | All items that match the selection criteria entered before EXCEPT are checked against *all* parameters contained within the parentheses following the keyword EXCEPT. If they match all these parameters too, they are not processed. | not applicable |

**Notes:**

1. Parameters that are irrelevant for OBJTYPE are ignored. For example: DATE, SIZE and USERID have no meaning for Natural error messages.
2. DBID, FNR, NAME, CIPHER and PASSWORD or PSW are ignored by the LOAD or SCAN command. These parameters must instead be specified in the *parameter-setting* clause as described for LOADFNAT... and LOADFUSER... in ***Keyword Explanation of*** *parameter-clause*.

## Natural-Related Debug Environment Selection

This selection is used to select Natural-related debug environments for processing.

The appropriate syntax is shown and explained in the following section.

- Syntax of Natural-Related Debug Environment Selection

### Syntax of Natural-Related Debug Environment Selection

```
object-name
NATPATH DEBUG
[LIBRARY library-name]
[     DBID dbid [FNR fnr] ]
[NAME vsam-name]
[CIPHER cipher]
[     { PASSWORD }        password ]
      { PSW      }
[ EXCEPT
      (object-name
      [LIBRARY library-name]
      )]
```

### Keyword Explanation of Natural-Related Debug Environment Selection

The keywords and valid input values for the debug environments to be processed are described in the following section.

| Keyword | Valid Values | Default Value |
|---|---|---|
| *object-name* | A valid debug environment name or a range of names.<br><br>See also ***Name*** in *Name, Date and Time Specification*. | none |
| LIBRARY | A valid library name or a range of names.<br><br>See also ***Name***. | none |
| DBID | A valid database ID. | 0<br>(current FUSER) |
| FNR | A valid file number. | 0<br>(current FUSER) |
| NAME | A valid VSAM name. | blank<br>(current FUSER) |
| CIPHER | The 8-digit cipher code of the Adabas file where the debug environments are stored. | blank<br>(current FUSER) |
| PASSWORD<br><br>or<br>PSW | An 8-character Adabas password. | blank<br>(current FUSER) |
| EXCEPT | See EXCEPT in *Natural Library Object and DDM Selection*. | not applicable |

**Note:** DBID, FNR, NAME, CIPHER and PASSWORD or PSW are ignored by the LOAD or SCAN command. These parameters must instead be specified in the *parameter-setting* clause as described for LOADFNAT... and LOADFUSER... in ***Keyword Explanation of*** *parameter-clause*.

## Natural-Related Profile Selection

This selection is used to select Natural-related profiles for processing.

The appropriate syntax is shown and explained in the following section.

- Syntax of Natural-Related Profile Selection

## Syntax of Natural-Related Profile Selection

```
object-name
NATPATH PROFILE
[OBJTYPE profile-type]
[    DBID dbid [FNR fnr] ]
[NAME vsam-name]
[CIPHER cipher]
[    { PASSWORD }    password    ]
     { PSW      }
[ EXCEPT
    (object-name
    [OBJTYPE profile-type]
    )]
```

### Keyword Explanation of Natural-Related Profile Selection

The keywords and valid input values for the profiles to be processed are described in the following section.

| Keyword | Valid Values | Default Value |
|---|---|---|
| *object-name* | A valid profile name or a range of names.<br><br>See also ***Name*** in *Name, Date and Time Specification*. | none |
| OBJTYPE | The type of profile:<br>D — Device profile<br>E — Editor profile<br>M — Map profile<br>P — Parameter profile<br>* — Asterisk (all profile types)<br>or any combination. | * |
| DBID | A valid database ID. | 0<br>(current FNAT) |
| FNR | A valid file number. | 0 |

| Keyword | Valid Values | Default Value |
|---|---|---|
| | | (current FNAT) |
| NAME | A valid VSAM name. | blank<br>(current FNAT) |
| CIPHER | The 8-digit cipher code of the Adabas file where the profiles are stored. | blank<br>(current FNAT) |
| PASSWORD<br>or<br>PSW | An 8-character Adabas password. | blank<br>(current FNAT) |
| | | |
| EXCEPT | See EXCEPT in *Natural Library Object and DDM Selection*. | not applicable |

**Note:** DBID, FNR, NAME, CIPHER and PASSWORD or PSW are ignored by the LOAD or SCAN command. These parameters must instead be specified in the *parameter-setting* clause as described for LOADFNAT... and LOADFUSER... in ***Keyword Explanation of*** *parameter-clause*.

## Natural-Related DL/I Subfile Selection

This selection is used to select Natural-Related DL/I subfiles for processing.

The appropriate syntax is shown and explained in the following section.

- Syntax of Natural-Related DL/I Subfile Selection

### Syntax of Natural-Related DL/I Subfile Selection

```
object-name
NATPATH SUBFILE
[OBJTYPE subfile-type]

[   DBID dbid [FNR fnr]   ]
[NAME vsam-name]
[CIPHER cipher]
[  { PASSWORD }          password  ]
   { PSW      }
[ EXCEPT
          ( object-name
```

```
            [OBJTYPE subfile-type]
            ) ]
```

### Keyword Explanation of Natural-Related DL/I Subfile Selection

The keywords and valid input values for the DL/I subfiles to be processed are described in the following section.

| Keyword | Valid Values | Default Value |
|---|---|---|
| *object-name* | A valid DL/I subfile name or a range of names.<br><br>See also ***Name*** in *Name, Date and Time Specification*. | none |
| OBJTYPE | The type of DL/I subfile:<br>`D` — NDB<br>`P` — NSB<br>`*` — Asterisk (both subfile types) | * |
| DBID | A valid database ID. | 0<br>(current FDIC) |
| FNR | A valid file number. | 0<br>(current FDIC) |
| NAME | A valid VSAM name. | blank<br>(current FDIC) |
| CIPHER | The 8-digit cipher code of the Adabas file where the DL/I subfiles are stored. | blank<br>(current FDIC) |
| PASSWORD<br><br>or<br>PSW | An 8-character Adabas password. | blank<br>(current FDIC) |
| EXCEPT | See EXCEPT in *Natural Library Object and DDM Selection*. | not applicable |

**Note:** DBID, FNR, NAME, CIPHER and PASSWORD or PSW are ignored by the LOAD or SCAN command. These parameters must instead be specified in the *parameter-setting* clause as described for LOADFNAT... and LOADFUSER... in ***Keyword Explanation of*** *parameter-clause*.

# Natural System Error Message Selection

This selection is used to select Natural system error messages for processing.

The appropriate syntax is shown and explained in the following section.

- Syntax of Natural System Error Message Selection

## Syntax of Natural System Error Message Selection

```
ERROR NATERROR
   DBID dbid FNR
[  fnr [NAME
   vsam-name]   [      {      PASSWORD }      password      ]      ]
   [CIPHER
   cipher]
[FMNUM error-number-from] [TONUM error-number-to]
[SLKIND message-type]
[LANGUAGE languages]
[ EXCEPT
   (
   [FMNUM error-number-from] [TONUM error-number-to]
   [SLKIND message-type]
   [LANGUAGE languages]
   ) ]
```

### Keyword Explanation of Natural System Error Message Selection

The keywords and valid input values for the Natural system error messages to be processed are described in the following section.

| Keyword | Valid Values | Default Value |
|---|---|---|
| DBID | Only applies to system error messages on mainframes.<br><br>A valid database ID. | 0<br>(current FNAT) |
| FNR | Only applies to system error messages on mainframes.<br><br>A valid file number. | 0<br>(current FNAT) |
| NAME | Only applies to system error messages on mainframes.<br><br>A valid VSAM name. | blank<br>(current FNAT) |

| Keyword | Valid Values | Default Value |
|---|---|---|
| CIPHER | Only applies to system error messages on mainframes.<br><br>The 8-digit cipher code of the Adabas file where the system error messages are stored. | blank<br>(current FNAT) |
| PASSWORD<br><br>or<br>PSW | Only applies to system error messages on mainframes.<br><br>An 8-character Adabas password. | blank<br>(current FNAT) |
| FMNUM | A start number of system error messages.<br><br>Valid range: 1 to 9999. | 1 |
| TONUM | An end number of system error messages.<br><br>Valid range: 1 to 9999.<br><br>The value must be greater than or equal to the value of FMNUM if specified. | 9999<br>or value of FMNUM<br>(if specified) |
| SLKIND | See SLKIND in *Natural Library Object and DDM Selection*. | A |
| LANGUAGE | Up to 8 valid language codes (for example, code 1 for English) of system error messages.<br><br>An asterisk (*) selects all language codes. | * |
| EXCEPT | See EXCEPT in *Natural Library Object and DDM Selection*. | |

**Note:** DBID, FNR, NAME, CIPHER and PASSWORD or PSW are ignored by the LOAD or SCAN command. These parameters must instead be specified in the *parameter-setting* clause as described for LOADFNAT... in ***Keyword Explanation of*** *parameter-clause*.

## Natural Command Processor Selection

This selection is used to select Natural command processor sources for processing.

The appropriate syntax is shown and explained in the following section.

- Syntax of Natural Command Processor Source Selection

## Syntax of Natural Command Processor Source Selection

```
object-name  PROCESSOR ncp-library-name
[            DBID ncp-dbid FNR ncp-fnr [file-options] ]
[EXCEPT
             (object-name
             [LIBRARY ncp-library-name]
             )]
```

*file-options*

```
[NAME ncp-vsam-name]
[CIPHER ncp-cipher]
[ { PASSWORD } ncp-password ]
  { PSW      }
```

**Note:** For the command FINDLIB, only the following keywords are processed: PROCESSOR, DBID, FNR, NAME, CIPHER and PASSWORD or PSW.

### Keyword Explanation of Natural Command Processor Source Selection

The keywords and valid input values for the Natural command processor sources to be processed are described in the following section.

| Keyword | Valid Values | Default Value |
|---|---|---|
| *object-name* | The name of a valid Natural command processor source or a range of names.<br><br>See also ***Name*** in *Name, Date and Time Specification*. | none |
| PROCESSOR | A valid library name or a range of names.<br><br>See also ***Name***. | none |
| DBID | The valid database ID of the Adabas file where the Natural command processor sources are stored. | 0 (current FNAT/FUSER) |
| FNR | The valid file number of the Adabas file where the Natural command processor sources are stored. | 0 (current FNAT/FUSER) |

| Keyword | Valid Values | Default Value |
|---|---|---|
| NAME | Only applies to Natural command processor sources on mainframes.<br><br>A valid VSAM name. | blank |
| CIPHER | The 8-digit cipher code of the Adabas file where the Natural command processor sources are stored. | blank |
| PASSWORD<br><br>or<br>PSW | The 8-character Adabas password of the Adabas file where the Natural command processor sources are stored. | blank |
| EXCEPT | See EXCEPT in *Natural Library Object and DDM Selection*. | |

**Note:** DBID, FNR, NAME, CIPHER and PASSWORD or PSW are ignored by the LOAD or SCAN command. These parameters must instead be specified in the *parameter-setting* clause as described for LOADNCP... in ***Keyword Explanation of*** *parameter-clause*.

## FDT Selection

This selection is used to select Adabas FDTs (Field Definition Tables) for processing.

For loading FDTs, see also ***FDTs*** in the section *Object Specification*.

The appropriate syntax is shown and explained in the following section.

- Syntax of FDT Selection

### Syntax of FDT Selection

```
FDT
DBID dbid
{ FNR fnr [CIPHER cipher] [ { PASSWORD } password ] }
{                         { PSW      }            }
{ FMFNR fnr-start TOFNR fnr-end                     }
```

### Keyword Explanation of FDT Selection

The keywords and valid input values for the FDTs to be processed are described in the following section.

| Keyword | Valid Values | Default Value |
|---|---|---|
| DBID | The database ID of the FDT. | none |
| FNR | The file number of the FDT. | none |
| CIPHER | The 8-digit Adabas cipher code of the FDT. | none |
| PASSWORD<br><br>or<br>PSW | The 8-character Adabas password of the FDT. | none |
| FMFNR | Only applies to the FIND or UNLOAD command.<br><br>A start value:<br>The file number (FNR) of an FDT. | none |
| TOFNR | Only applies to the FIND or UNLOAD command.<br><br>An end value:<br>The file number (FNR) of an FDT. | none |

## Application Selection

This selection applies to applications created and maintained in Natural Studio's application workspace and the libraries or objects that belong to these applications.

The appropriate syntax is shown and explained in the following section.

- Selecting Base and Compound Applications
- Selecting Application Libraries
- Selecting Application Objects

### Selecting Base and Compound Applications

This selection only applies to the find function.

**Syntax**

```
APPLICATION APNAME application-name
[APTYPE application-type]
[COMPAPPLICATION compound-application-name]
[                    EXCEPT
                     (APNAME application-name
                     [APTYPE application-type]
                     ) ]
```

### Selecting Application Libraries

This selection only applies to the find function.

**Syntax**

```
APPLICATION  APLIBRARY application-library-name
[BASEAPPLICATION base-application-name]
[ COMPAPPLICATION compound-application-name]
[            DBID dbid [FNR fnr] ]
[            EXCEPT
             (APLIBRARY application-library-name
             [BASEAPPLICATION base-application-name]
             ) ]
```

### Selecting Application Objects

This selection only applies to the find and unload functions.

**Syntax**

```
APPLICATION  APOBJECTS application-object-name
[BASEAPPLICATION base-application-name]
[COMPAPPLICATION compound-application-name]
[LIBRARY library-name]
[object-specification]
[                        EXCEPT
                         (APOBJECT application-object-name
                         [LIBRARY library-name]
                         [BASEAPPLICATION base-application-name]
                         [object-specification]
                         ) ]
```

**Keyword Explanation of Application Selection**

The keywords and valid input values for the applications, application libraries or application objects to be processed are described in the following section.

| Keyword | Valid Values | Default Value |
|---|---|---|
| APNAME | A valid name of a Natural application or a range of names.<br><br>See also ***Name*** in *Name, Date and Time Specification*. | * |
| APTYPE | A valid application type:<br>B — Base application<br>O — Compound application<br>* — All: base and/or compound applications | * |
| COMPAPPLICATION | Only applies if APTYPE is set to * or B.<br><br>The name of a compound application to which the specified base application belongs or a range of names.<br><br>Only base applications that belong to the specified compound application(s) are selected; base applications that do not belong to a compound application are not selected. | none |
| EXCEPT | See EXCEPT in *Natural Library Object and DDM Selection*. | not applicable |
| APLIBRARY | The valid name of a library that belongs to a Natural base or compound application or a range of names.<br><br>See also ***Name*** in *Name, Date and Time Specification*. | * |
| BASEAPPLICATION | The valid name of a Natural base application to which an application library or application object belongs.<br><br>See also ***Name*** in *Name, Date and Time Specification*. | * |
| DBID | The valid database ID of an application library. | 0<br>(no check) |
| FNR | The valid file number of an application library. | 0<br>(no check) |
| APOBJECT | The valid name of an application object that belongs to a base or compound application, or a range of names.<br><br>See also ***Name*** in *Name, Date and Time Specification*. | * |
| LIBRARY | A valid library name or a range of names.<br><br>If OBJTYPE is set to D (see *Natural Library Object and DDM Selection*), the library name is ignored.<br><br>See also ***Name*** in *Name, Date and Time Specification*. | * |
| *object-specification* | Indicates that additional selection criteria can be specified for application objects as shown in the **syntax** diagram for Natural library objects and DDMs: all items listed below LIBRARY *library-name* can also be applied to application objects whereas | not applicable |

| Keyword | Valid Values | Default Value |
|---|---|---|
| | *object-name* in the EXCEPT clause is irrelevant for application objects. | |

## Object Selection for Delete Instructions

This selection is used to specify delete instructions for Natural library objects, DDMs, user-defined error messages and Natural system error messages. The delete instructions are executed when a work file of internal format is loaded in the target environment with the DELETEALLOWED option specified.

The appropriate syntax is shown and explained in the following section.

- Syntax of Delete Instructions for Natural Library Objects and DDMs
- Syntax of Delete Instructions for User-Defined Error Messages
- Syntax of Delete Instructions for Natural System Error Messages

### Syntax of Delete Instructions for Natural Library Objects and DDMs

```
object-name
LIBRARY library-name
[ OBJTYPE { N } ]
          { D }
[ NATTYPE { * } ]
[SCKIND object-kind]
```

#### Keyword Explanation of Delete Instructions for Natural Library Objects and DDMs

The keywords and valid values for the objects to be processed are described in the following section.

| Keyword | Valid Values | Default Value |
|---|---|---|
| *object-name* | A valid object name or a start value (*value**) for a range of names such as ABC*. | none |
| LIBRARY | A valid library name. A range specification is *not* allowed.<br><br>If OBJTYPE (see below) is set to D, the library name is ignored. | none |

| Keyword | Valid Values | Default Value |
|---|---|---|
| OBJTYPE | A valid object-type code:<br><br>D DDMs<br>N Natural programming objects<br><br>Object type D cannot be combined with object type N. | * |
| NATTYPE | Not applicable if OBJTYPE is set to D.<br><br>A Natural object type. Valid input values are:<br><br>* All object types | * |
| SCKIND | Not applicable if OBJTYPE is set to D.<br><br>The kind of Natural programming objects.<br>Valid input values are:<br><br>S Source objects. If used in the *except-clause* (see *Syntax of Natural Library Object and DDM Selection*): objects that are stored only in source form.<br>C Cataloged objects. If used in the *except-clause*: objects that are stored only in cataloged form.<br>A All source and cataloged objects. | A |

### Syntax of Delete Instructions for User-Defined Error Messages

```
*
LIBRARY library-name
OBJTYPE E
FMNUM error-number-from
[TONUM error-number-to]
[SLKIND message-type]
[LANGUAGE languages]
```

*library-name* denotes the name of a single library; a range specification is not allowed.

For explanations of the other elements used in this syntax, see ***Keyword Explanation of Natural Library Object and DDM Selection***.

### Syntax of Delete Instructions for Natural System Error Messages

```
ERROR NATERROR
FMNUM error-number-from
[TONUM error-number-to]
[SLKIND message-type]
[LANGUAGE languages]
```

For explanations of the elements used in this syntax, see ***Keyword Explanation of Natural System Error Message Selection***.

## Help Text Selection

This selection is used to specify that Natural help texts are processed during the unload, load, scan and find functions when internal format is used for the work files. These help texts are identical to the help information provided by the Natural Help utility, which is invoked with the `HELP` system command (see the *System Commands* documentation).

### Syntax of Help Text Selection

```
ERROR NATURAL HELP
[ DBID dbid FNR fnr [NAME vsam-name]   [ { PASSWORD } password ] ]
  [CIPHER cipher]                        { PSW      }
```

For explanations of the elements used in this syntax, see ***Keyword Explanation of Natural Library Object and DDM Selection***.

# 41 Object List - LIST Workplan


- Syntax of object-type-and-location ..... 248
- Syntax of object-name-description ..... 250
- Example of an Object List ..... 252

An object list is a Workplan of the type LIST, which specifies object selection criteria for the objects to be processed in the `UNLOAD`, `LOAD`, `FIND` or `DELETE` command. An object list can be used as an alternative to the *`select-clause`* and the SELECTION Workplan.

The following syntax applies to an object list:

```
TYPE LIST
{ object-type-and-location (object-name-description ...) } ...
```

The syntactical options are explained in the following section. The keywords and variable values contained in the syntax diagrams shown in this section represent parameters that are used to specify object selection criteria. If indicated, a variable value must be supplied with a keyword. Each syntax element (except for the ones enclosed in parentheses) must start on a new line and end on the same line.

For explanations of the keywords contained in the syntax diagrams, refer to the section *`select-clause`*.

## Syntax of object-type-and-location

The syntax diagrams that apply to *`object-type-and-location`* are shown in the following section.

- Natural Objects and DDMs
- Natural System Error Messages
- Natural Command Processor Sources
- Natural-Related Debug Environments
- Natural-Related Profiles
- Natural-Related DL/I Subfiles
- FDTs

### Natural Objects and DDMs

```
LIBRARY library-name
[ DBID dbid FNR fnr [NAME vsam-name] [CIPHER cipher]  [ { PASSWORD } password ] ]
                                                     { PSW }
[ OBJTYPE group-type]
```

**Notes:**

1. No ranges are allowed for *`library-name`*.

2. For DDMs,`OBJTYPE` must be set to `D`.

### Natural System Error Messages

```
ERROR NATERROR
[ DBID dbid FNR fnr [NAME vsam-name] [CIPHER cipher] [ { PASSWORD } password ] ]
                                                   { PSW      }
```

### Natural Command Processor Sources

```
PROCESSOR ncp-library-name
[ DBID dbid FNR fnr [NAME vsam-name] [CIPHER cipher] [ { PASSWORD } password ] ]
                                                     { PSW      }
```

**Note:** No ranges are allowed for *ncp-library-name*.

### Natural-Related Debug Environments

```
NATPATH DEBUG
LIBRARY library-name
[ [ DBID dbid [FNR fnr] ] [NAME vsam-name] [CIPHER cipher] { PASSWORD } password ] ]
                                                           { PSW      }
```

### Natural-Related Profiles

```
NATPATH PROFILE
[ [ DBID dbid [FNR fnr] ] [NAME vsam-name] [CIPHER cipher] [ { PASSWORD } password ] ]
                                                            { PSW      }
```

### Natural-Related DL/I Subfiles

```
NATPATH SUBFILE
[ [ DBID dbid [FNR fnr] ] [NAME vsam-name] [CIPHER cipher] [ { PASSWORD | PSW } password ] ]
```

### FDTs

```
FDT
```

## Syntax of object-name-description

The syntax diagrams that apply to *object-name-description* are shown in the following section:

- Natural Objects
- DDMs
- Natural System Error Messages
- Natural Command Processor Sources
- Natural-Related Debug Environments
- Natural-Related Profiles
- Natural-Related DL/I Subfiles
- FDTs

### Natural Objects

```
{ object-name [SCKIND object-kind]
  error-number [ SLKIND message-type] [ LANGUAGE languages]
  FMNUM error-number-from TONUM error-number-to [SLKIND message-type] [LANGUAGE languages] }
```

### DDMs

```
object-name
```

### Natural System Error Messages

```
{ error-number [ SLKIND message-type] [ LANGUAGE languages]
  FMNUM error-number-from TONUM error-number-to [ SLKIND message-type] [ LANGUAGE
  languages] }
```

### Natural Command Processor Sources

```
object-name
```

### Natural-Related Debug Environments

```
object-name
```

### Natural-Related Profiles

```
object-name [ OBJTYPE profile-type]
```

### Natural-Related DL/I Subfiles

```
object-name [ OBJTYPE subfile-type]
```

### FDTs

```
DBID dbid FNR fnr [CIPHER cipher] [ { PASSWORD } password ]
                                    { PSW      }
```

## Example of an Object List

The follwing is an example of a Workplan of the type LIST:

```
TYPE LIST
  LIBRARY LIB-1 OBJTYPE N    /* process Natural objects from library 'LIB-1'
   ( A* SCKIND S             /* all sources objects whose names start with 'A'
  B1                         /* source and/or cataloged object of  'B1'
  CDE> SCKIND C )           /* all cataloged objects with names greater than/equal ↩
to 'CDE'
  /*                         /* comment line
  LIBRARY LIB-2              /* process Natural objects from library 'LIB-2'
                             /* including error messages and shared resources
  ( *                        /* all source and/or cataloged objects
                             /* including shared resources
  FMNUM 1 TONUM 100          /* error messages from 1 to 100
  )
```

# 42 parameter-setting


- Syntax of parameter-clause ..... 254
- Keyword Explanation of parameter-clause ..... 255

The *parameter-setting* clause is used to change attributes for the LOAD or UNLOAD command for the objects to be processed and to define target destinations for the LOAD command (for example, FNAT).

The following syntax applies to the *parameter-setting* clause:

```
WITH
{ (parameter-workplan)  }
{ parameter-clause      }
```

For an explanation of the syntax that applies to *parameter-workplan*, refer to ***Referencing Workplans*** in the section *Workplans*.

This section covers the following topics:

## Syntax of parameter-clause

The syntax of the *parameter-clause* is shown in the following diagram. If indicated, a variable value must be supplied with a keyword.

```
[      [NAME old-name] NEWNAME new-name ]
[      [LIBRARY old-library-name]       ]
       NEWLIBRARY new-library-name
[      LOADFNATDBID fnat-dbid LOADFNATFNR fnat-fnr                    ]
       [LOADFNATNAME vsam-name]
       [LOADFNATCIPHER fnat-cipher]
[ [ { LOADFNATPASSWORD } fnat-password ] ]
  [ { LOADFNATPSW      }               ]
[      LOADFUSERDBID fuser-dbid LOADFUSERFNR fuser-fnr                ]
       [LOADFUSERNAME fuser-vsam-name]
       [LOADFUSERCIPHER fuser-cipher]
[ [ { LOADFUSERPASSWORD } fuser-password ] ]
  [ { LOADFUSERPSW      }                ]
[      LOADNCPDBID ncp-file-dbid LOADNCPFNR ncp-file-fnr              ]
       [LOADNCPNAME ncp-file-vsam-name]
       [LOADNCPCIPHER ncp-file-cipher]
[ [ { LOADNCPPASSWORD } ncp-file-password ] ]
  [ { LOADNCPPSW      }                   ]
```

```
[       [FDTDBID old-fdt-dbid FDTFNR old-fdt-fnr] NEWFDTDBID new-fdt-dbid NEWFDTFNR      ]
        new-fdt-fnr
[ERRNUMDIFF modification-of-error-message-range]
[       [LANGUAGE old-language]                                                          ]
        NEWLANGUAGE new-language
[       [DATE old-date] NEWDATE new-date                                                 ]
[       [USERID old-userid]  NEWUSERID                                                   ]
        new-userid
[       [TID old-terminal-id] NEWTID                                                     ]
        new-terminal-id
[       [PATH old-external-path-name]                                                    ]
        NEWPATH new-external-path-name
```

## Keyword Explanation of parameter-clause

The keywords and variable values (if relevant) of the *parameter-clause* are explained in the following section.

| Keyword | Values | Restricted to Command |
|---|---|---|
| NAME | The object name to be checked if NEWNAME is specified. | |
| NEWNAME | A new object name.<br><br>**Note:** Not applicable to DDMs on mainframe platforms. | |
| LIBRARY | The library name to be checked if NEWLIBRARY is specified. | |
| NEWLIBRARY | A new library name.<br><br>**Note for the LOAD function:**<br><br>NEWLIBRARY does *not* affect the library name used in the delete instruction of a work file that is processed with the DELETEALLOWED option. | |
| LOADFNATDBID | The database ID (DBID) of FNAT libraries. | LOAD |
| LOADFNATFNR | The file number (FNR) of FNAT libraries. | LOAD |
| LOADFNATNAME | Only applies to objects on mainframes.<br>An FNAT VSAM file name. | LOAD |
| LOADFNATCIPHER | Only applies to objects on mainframes.<br>An FNAT cipher code. | LOAD |
| LOADFNATPASSWORD<br><br>or | Only applies to objects on mainframes.<br>An FNAT Adabas password. | LOAD |

| Keyword | Values | Restricted to Command |
|---|---|---|
| LOADFNATPSW | | |
| LOADFUSERDBID | The DBID of FUSER libraries. | LOAD |
| LOADFUSERFNR | The FNR of FUSER libraries. | LOAD |
| LOADFUSERNAME | Only applies to objects on mainframes.<br>An FUSER VSAM file name. | LOAD |
| LOADFUSERCIPHER | Only applies to objects on mainframes.<br>An FUSER cipher code. | LOAD |
| LOADFUSERPASSWORD<br><br>or<br>LOADFUSERPSW | Only applies to objects on mainframes.<br>An FUSER Adabas password. | LOAD |
| LOADNCPDBID | The DBID of the Adabas file for Natural command processor sources. | LOAD |
| LOADNCPFNR | The FNR of the Adabas file for Natural command processor sources. | LOAD |
| LOADNCPNAME | Only applies to objects on mainframes.<br>The VSAM name of the Adabas file for Natural command processor sources. | LOAD |
| LOADNCPCIPHER | The cipher code of the Adabas file for Natural command processor sources. | LOAD |
| LOADNCPPASSWORD<br><br>or<br>LOADNCPPSW | Only applies to objects on mainframes.<br>The Adabas password of the Adabas file for Natural command processor sources. | LOAD |
| FDTDBID | The DBID of the Adabas FDT (Field Definition Table) to be checked if NEWFDTDBID is specified. | |
| NEWFDTDBID | A new DBID of the FDT. | |
| FDTFNR | The DBID of the FDT to be checked if NEWFDTFNR is specified. | |
| NEWFDTFNR | A new FNR of the FDT. | |
| ERRNUMDIFF | A number (positive or negative) that is to be added to the Natural error messages during the UNLOAD or LOAD command.<br><br>ERRNUMDIFF can only be specified if FMNUM and TONUM (see *select-clause*) have been specified as selection criteria. Otherwise, it is not possible to check for valid results. | |
| LANGUAGE | Up to 8 valid language codes (for example, code 1 for English) of Natural error messages to be checked if NEWLANGUAGE (see below) is specified.<br><br>If *language* contains more than one language code, *new-language* must contain the same numbers of language codes. Each *language* language code is replaced by the language code in the corresponding position of *new-language*. | |

| Keyword | Values | Restricted to Command |
|---|---|---|
| | If *language* is not specified, *new-language* must not contain more than one language code. | |
| NEWLANGUAGE | Up to 8 valid language codes (for example, code 4 for Spanish) for new user-defined error messages. This option does not apply to the long texts of Natural system error messages for which English (language code 1) is the only valid language.<br><br>See also LANGUAGE above. | |
| DATE | An object date.<br><br>You can add a time by inserting a blank between date and time. For the format and ranges allowed, see ***Date*** and ***Time*** in *Name, Date and Time Specification*. | |
| NEWDATE | A new object date.<br><br>NEWDATE can be a date followed by a time value. You can add a time by inserting a blank between date and time. See also ***Date*** and ***Time*** in *Name, Date and Time Specification*. | |
| USERID | The user ID to be checked if NEWUSERID is specified. | |
| NEWUSERID | A new user ID. | |
| TID | Only applies to objects on mainframes.<br>The terminal ID to be checked if NEWTID is specified. | |
| NEWTID | Only applies to objects on mainframes.<br>A new terminal ID. | |
| PATH | The path name to be checked if NEWPATH is specified. | |
| NEWPATH | A new path name. | |

**Notes:**

1. Parameters not applicable to the selection criterion processed are ignored.
2. LOADFNAT..., LOADFUSER... and LOADNCP... are used for the LOAD command only, and ignored otherwise.
3. LOADFNAT... is used for libraries starting with SYS (except SYSTEM).
4. LOADFUSER... is used for libraries not starting with SYS (but including SYSTEM).
5. LOADNCP... is used for Natural command processor sources.

# 43 option-setting


- Syntax of option-setting ..... 260
- Keyword Explanation of option-setting ..... 262

The *option-setting* clause is used to change the default values of Object Handler command options.

The syntax that applies to the *option-setting* clause is shown and explained in the following section. The keywords and variable values contained in the syntax diagrams shown represent the parameters that are used to specify the default values. If indicated, a variable value must be supplied with a keyword.

## Syntax of option-setting

```
WHERE
{  (option-workplan)  }
{  option-clause      }
```

The syntax diagram that applies to *option-workplan* is shown and described in ***Referencing Workplans*** in the section *Workplans*.

The syntax of the *option-clause* is shown in the following section.

- Syntax of option-clause

### Syntax of option-clause

```
[ REPLACE { ALL      } ]
          { OBSOLETE }
          { EXCEPT   }

[ transfer-options          ]
  internal-format-options

[ NOREPORT                  ]
  NEWREPORT [file-name]
  REPORT [file-name]
  BATCHREPORT

[ {REPORT-LIBRARY|REP-LIB} library-name
  [REP-LIB-DBID dbid [REP-LIB-FNR fnr]] [REP-LIB-NAME vsam-name]
  [{REP-LIB-PASSWORD|REP-LIB-PSW} password] [REP-LIB-CIPHER cipher]
]

[ REPORT-OPTION-1 { A } ]
                  { E }
                  { S }
```

```
[ REPORT-FORMAT { S } ]
                { T }

[ REPORT-MODE { S } ]
              { L }

[ NORESTART                              ]
  RESTART [restart-text-object]

[NUMBERPROCESS number]

[FIXEDLENGTH]

[FDIC (dbid,fnr,password,cipher)]

[FSEC (dbid,fnr,password,cipher)]

[ { WORKFILETYPE } { DEFAULT } ]
  { WFTYPE       } { UTF-8   }

[ { PC    } [file-name] ]
  { NEWPC }

[ { PCCOMMANDFILENUMBER } command-file-number ]
  { PCCFN               }
```

**Separators**

Commas must be used as separators between the values following the `FDIC` and `FSEC` keywords, or if a value is missing. For example: `FDIC (10,21,,2a)`.

If the session parameter `ID` (see *ID - Input Delimiter Character* in the *Parameter Reference* documentation) has been set to a comma, use a slash (/) as the separator between values.

*transfer-options*

```
TRANSFER

[ CONVERSION-TABLE { SYSTEM-TABLE                } ]
                   { USER-TABLE                  }
                   { [conversion-program]        }

[SUBSTITUTE]

[INCLUDE-LINE-NUMBERS]

[UPPERCASE-TRANSLATION]

[INCORPORATE-FREE-RULES]

[LOAD-CODE-PAGE code-page-name]

[DA-FORMAT data-area-format]

[UNSHAPE]
```

```
[USE-LINE-NUMBER-INCREMENT]
```

*internal-format-options*

```
[        { ON      }  ]
  XREF   { OFF     }
         { DOC     }
         { FORCE   }
         { SPECIAL }  ]
[DELETEALLOWED]
[NOSYMBOLTABLE]
[VERSIONCHECK]
```

## Keyword Explanation of option-setting

The keywords and the variable values (if relevant) of *option-setting* are explained in the following section:

| Option | Explanation | Restricted to Command |
|---|---|---|
| REPLACE | Replaces existing objects according to the option specified:<br>ALL — All objects (default setting).<br>OBSOLETE — All objects with a date older than the date of the object in the load file.<br>EXCEPT — All objects except those with a date newer than the date of the object in the load file. | LOAD<br>LOADALL |
| TRANSFER | Set Transfer mode.<br>The data is read and written in Transfer format. For valid options, see ***Keyword Explanation of transfer-options***. | UNLOAD<br>LOAD<br>SCAN |
| NOREPORT | Specifies the report file setting:<br>No data is recorded to a report file. This is the default setting for the FIND and FINDLIB commands. | |
| NEWREPORT | Specifies the report file setting:<br>Report data is recorded and written to a Natural text member stored in the Workplan library. An existing file will be overwritten. | |

| Option | Explanation | Restricted to Command |
|---|---|---|
| REPORT | Specifies the report file setting:<br><br>Report data is recorded and written to a Natural text member stored in the Workplan library. This is the default setting for the commands UNLOAD, LOAD, LOADALL, SCAN, SCANALL and DELETE.<br><br>By default, this Natural text member is deleted when the Object Handler is terminated. To change default settings, see the documentation on profile parameter **DELETE-TEMPORARY-REPORT-TEXT**. | |
| BATCHREPORT | Specifies the report setting for batch processing or when using the OBJHAPI Application Programming Interface:<br><br>Report data is either written to SYSOUT or output on the screen respectively (report data is *not* written to a file). | |
| REPORT-LIBRARY<br>or<br>REPORT-LIB<br>or<br>REP-LIB | Specifies the library where the report text object is stored if the option REPORT or NEWREPORT is specified.<br><br>If this option is not specified, the library WORKPLAN or the library in the Object Handler profile parameter Workplan-Library is used. See also ***Profile Parameters***. | |
| REP-LIB-DBID | The database ID of the system file where the REPORT-LIBRARY resides.<br><br>(0 is the current FNAT/FUSER.) | |
| REP-LIB-FNR | The file number of the system file where the REPORT-LIBRARY resides.<br><br>(0 is the current FNAT/FUSER.) | |
| REP-LIB-NAME | The VSAM name of the system file where the REPORT-LIBRARY resides. | |
| REP-LIB-PASSWORD<br>or<br>REP-LIB-PSW | The Adabas password of the system file where the REPORT-LIBRARY resides. | |
| REP-LIB-CIPHER | The Adabas cipher code of the system file where the REPORT-LIBRARY resides. | |
| REPORT-OPTION-1<br>or<br>REPOPT1<br>or<br>REP-OPT-1 | Specifies the report option to be used when a direct command is executed and a report is to be written:<br><br>A Display all report items (default).<br>E Display only error messages. This includes messages from Natural Security and messages that have incurred during the execution of a LOAD command, for instance “not replaced”.<br>S For batch mode only. The error report is split into two parts: Report items except error messages are written to the default | UNLOAD<br>LOAD<br>SCAN<br>DELETE |

| Option | Explanation | Restricted to Command |
|---|---|---|
| | report device (REPORT(0)/CMPRINT), error messages (including messages from Natural Security and messages that have incurred during the execution of a LOAD command) are written to the second report device (REPORT(1)/CMPRT01). Note that in online mode S has the same effect as A. | |
| REPORT-FORMAT<br>or<br>REPFRMT | Specifies the report format to be used when a direct command is executed and a report is to be written:<br><br>S Display the source data in the unload report, i.e. the data of the unloaded object before parameters (e.g. the library name) are changed (default setting).<br>T Display the target data in the unload report, i.e. the data after parameters (e.g. the library name) have been changed. | UNLOAD |
| REPORT-MODE<br>or<br>REPM | Specifies the report mode to be used when a direct command is executed and a report is to be written:<br><br>S Write a short report, i.e. the most relevant data is displayed on the first 80 columns of the report line with short delimiters.<br>L Write a large report, i.e. the data is displayed in the original order with large delimiters (default setting). | |
| NORESTART | No restart information is written to a file. | LOAD |
| RESTART | Restart information is written to a Natural text member stored in the Workplan library. | LOAD |
| NUMBERPROCESS | Specifies the number of objects to be processed.<br><br>The LOAD or SCAN command stops execution after the number specified. | LOAD<br>SCAN |
| FIXEDLENGTH | Sets the format of the unload work file to a maximum record length of fixed size.<br><br>Every data record contains 256 bytes if written in internal format, or 100 bytes in Transfer format. | UNLOAD |
| FDIC | Specifies the system file FDIC to be used for processing:<br><br>the database ID (*dbid*), file number (*fnr*), password (*password*) and cipher code (*cipher*) of the Adabas file.<br><br>If no values (or 0) are specified, the current FDIC system file is used. | UNLOAD<br>LOAD<br>DELETE |
| FSEC | Specifies the system file FSEC to be used for processing:<br><br>the database ID (*dbid*), file number (*fnr*), password (*password*) and cipher code (*cipher*) of the Adabas file. | UNLOAD<br>LOAD<br>DELETE |

| Option | Explanation | Restricted to Command |
|---|---|---|
| | If no values (or 0) are specified, the current FSEC system file is used. | |
| WORKFILETYPE<br>or<br>WFTYPE | The work file type of Natural Work File 1 when data is read and written to the work file:<br><br>DEFAULT Default binary work file.<br>UTF-8 Unicode/UTF-F8 encoded binary work file.<br>UTF-8 only applies to the unload function and if TRANSFER is specified.<br>If UTF-8 is specified, you cannot use the options CONVERSION-TABLE, SUBSTITUTE and INCORPORATE-FREE-RULES.<br><br>If WORKFILETYPE has not been specified, the current type is used. | UNLOAD |
| PC<br>NEWPC | Only applies if Entire Connection is installed.<br><br>Writes data to or reads data from an Entire Connection work file. *file-name* denotes the complete path name assigned to the Entire Connection work file. If your system environment does not accept a backslash (\) separator, use a slash (/) instead. If you do not specify *file-name*, Entire Connection prompts you for the name of a work file.<br><br>If NEWPC is specified, the data unloaded overwrites the contents of the existing work file or fills a new work file from the top. Otherwise, the data is appended.<br><br>See also ***Work File Assignment*** in *Work Files*. | UNLOAD<br>LOAD<br>SCAN |
| PCCOMMANDFILENUMBER<br>or<br>PCCOM<br>or<br>PCCFN | Only applies if Entire Connection is installed.<br><br>Specifies the number of the work file that is used for processing Entire Connection commands.<br><br>The default value is 8 for Work File 8, which must be defined as Entire Connection work file.<br><br>See also ***Work File Assignment*** in *Work Files*. | UNLOAD<br>LOAD<br>SCAN |

The keywords and the variable values (if relevant) of *transfer-options* and *internal-format-options* are explained in the following section:

- Keyword Explanation of transfer-options

- Keyword Explanation of internal-format-options

### Keyword Explanation of transfer-options

When using the `TRANSFER` keyword, you can specify the following options:

| Option | Explanation | Restricted to Command |
|---|---|---|
| `CONVERSION-TABLE` | Converts data processed in Transfer format by using either of the following conversion tables:<br><br>`SYSTEM-TABLE`:<br><br>The internal Natural conversion table.<br><br>`USER-TABLE`:<br><br>A user-defined conversion table if *`conversion-program`* has been specified. This program must be stored in the library SYSOBJH or one of its steplibs; see the example programs OTNCONAE and OTNCONEA in the library SYSOBJH. | `UNLOAD`<br>`LOAD`<br>`SCAN` |
| `SUBSTITUTE` | Replaces line references by labels during the unload in Transfer format.<br><br>This option only applies if your source-code line numbers are used for statement references. If so, the line numbers of referenced lines and the line number references are replaced by labels. The sources are not modified in the database. | `UNLOAD` |
| `INCLUDE-LINE-NUMBERS` | Transfers line numbers during the unload in Transfer format.<br>By default, line numbers in Natural objects are *not* unloaded. | `UNLOAD` |
| `USE-LINE-NUMBER-INCREMENT`<br>or<br>`USE-LNI` | `UNLOAD` If the option `INCLUDE-LINE-NUMBERS` is not specified, the line number increment of Natural source objects will be unloaded. By default, the line number increment in Natural source objects is *not* unloaded.<br><br>`LOAD` If the line number increment was transferred, it is used to rebuild the line numbers of the Natural source objects. | `UNLOAD`<br>`LOAD` |
| `UPPERCASE-TRANSLATION` | Translates any source code into upper case during the load in Transfer format.<br>By default, source code in Natural objects is *not* translated. | `LOAD` |
| `INCORPORATE-FREE-RULES` | Incorporates source text of Predict free rules associated with a map into a map source during the unload in Transfer format if Predict is installed. | `UNLOAD` |

| Option | Explanation | Restricted to Command |
|---|---|---|
| LOAD-CODE-PAGE | Specifies the code page to be used for converting object sources encoded in Unicode/UTF-8 (Universal Transformation Format, 8-bit form).<br><br>If you use this option, all object sources unloaded into a work file in UTF-8, will be converted with the specified code page when they are loaded into a work file.<br><br>If you specify *CODEPAGE as *code-page-name* or if *code-page-name* is not specified, the value assigned to the system variable *CODEPAGE is used (see the *System Variables* documentation).<br><br>If LOAD-CODE-PAGE is specified, you cannot use the options CONVERSION-TABLE and UPPERCASE-TRANSLATION. | LOAD<br>LOADALL |
| DA-FORMAT | Specifies format conversion of data area sources: see **Data area format** in *Transfer Options* in *Settings*. | UNLOAD<br>LOAD |
| UNSHAPE | Replaces shaped Arabic characters (code page IBM420) with the corresponding unshaped characters in the following source objects: Natural programs , user error messages and system error messages. | UNLOAD<br>LOAD |

### Keyword Explanation of internal-format-options

When using *internal-format-options*, you can specify the following:

| Option | Explanation | | Restricted to Command |
|---|---|---|---|
| XREF | Only applies if Predict is installed.<br><br>Loads or unloads XRef data of cataloged Natural objects. You can specify one of the following values: | | LOAD<br>UNLOAD |
| | | | |
| | ON | UNLOAD:<br>Unloads cataloged objects and their cross-reference data (if any).<br><br>LOAD:<br>Loads cataloged objects and their cross-reference data if cross-references exist in the work file. | |
| | | | |
| | OFF | No XRef data is processed. This is the default. | |
| | | | |

| Option | Explanation | | Restricted to Command |
|---|---|---|---|
| | DOC | Only applies to LOAD.<br><br>Loads cataloged objects and their cross-reference data (if any) only if Predict entries exist for the objects in the FDIC system file. | |
| | | | |
| | FORCE | Only applies to LOAD.<br><br>Loads cataloged objects and their cross-reference data only if cross-references exist in the work file and if Predict entries exist for the objects in the FDIC system file. | |
| | | | |
| | SPECIAL | Only applies to LOAD.<br><br>Loads cataloged objects and their cross-reference data (if any). | |
| DELETEALLOWED | Processes delete instructions from work files when loading objects in internal format. | | LOAD |
| NOSYMBOLTABLE | Unloads cataloged Natural library objects without their corresponding internal Natural symbol tables.<br><br>This will reduce the amount of disk storage required. However, this is only useful for a production environment, as several application development functions which require the symbol tables will then not be available; in addition, the profile parameter RECAT=ON (see the *Parameter Reference* documentation) will not apply. | | UNLOAD |
| VERSIONCHECK | Checks the Natural version of the cataloged object to be loaded. The Natural version under which the objects were cataloged and written to the work file is compared with the current Natural version. Objects cataloged under a Natural version higher than the current one will be rejected.<br><br>VERSIONCHECK can only be used if data is loaded in internal format, that is, if the TRANSFER option is *not* specified. | | LOAD |

# 44 Examples of Using Direct Commands


- Unloading Objects for the Same Platform ..... 270
- Unloading Objects for Different Platforms ..... 271
- Loading Objects in Internal Format ..... 271
- Loading Objects in Transfer Format ..... 272

This section provides examples for using Object Handler direct commands.

**Tip:** For additional examples, you can view the command generated for an Object Handler function. This command is automatically displayed when you use a wizard. In advanced-user mode, you can activate the display of the command by either entering the Object Handler command `SET ADVANCEDCMD ON` or setting the parameter `Display-Cmd-in-Advanced-Mode` to `Y` (Yes) in the Object Handler profile (see also *Profile Settings*).

## Unloading Objects for the Same Platform

This section contains examples of how to unload objects in internal format to a work file in order to load them on the same platform, within either a local mainframe, UNIX, OpenVMS or Windows environment:

- Unload all Natural programming objects (source objects only) from library `ABC`:

```
UNLOAD * LIB ABC OBJTYPE N SCKIND S
```

- Unload all Natural programming objects (cataloged objects only) from library `ABC`:

```
UNLOAD * LIB ABC OBJTYPE N SCKIND C
```

- Unload all Natural programming objects (cataloged objects and source objects) from library `ABC`:

```
UNLOAD * LIB ABC OBJTYPE N SCKIND A
```

- Unload all Natural programming objects (source objects only) from library `ABC` to load in library `ABCNEW`:

```
UNLOAD * LIB ABC OBJTYPE N SCKIND S WITH NEWLIBRARY ABCNEW
```

- On a mainframe: Unload all DDMs whose names start with `EMP` and which point to database `88`:

```
UNLOAD EMP* LIB * OBJTYPE D DDMDBID 88
```

- On UNIX, OpenVMS or Windows: Unload all DDMs whose names start with `EMP` and which point to database `88`:

```
UNLOAD EMP* LIB * OBJTYPE N NATTYPE V DDMDBID 88
```

- On UNIX, OpenVMS or Windows: Unload all DDMs whose names start with `EMP` from library `VLIB` to load in library `VLIBNEW`:

```
UNLOAD EMP* LIB VLIB OBJTYPE N NATTYPE V WITH NEWLIBRARY VLIBNEW
```

- Unload all user-defined error messages from library `ERRLIB` to load in library `NEWERR`:

```
UNLOAD * LIB ERRLIB OBJTYPE E SLKIND A WITH NEWLIBRARY NEWERR
```

- On Windows: Unload all Natural programming objects (cataloged objects and source objects) from library `ABC` to a portable work file on a PC:

```
UNLOAD * LIB ABC OBJTYPE N WHERE WORKFILE C:\WF1.SAG WORKFILETYPE PORTABLE
```

  or

```
UNLOAD * LIB ABC OBJTYPE N WHERE WORK C:\WF1.SAG WFT P
```

## Unloading Objects for Different Platforms

This section contains command examples of how to unload objects in Transfer format to a work file in order to load them on a different platform such as unloading in a mainframe and loading in a UNIX, an OpenVMS or a Windows environment.

- Unload all Natural programming objects (source objects only) from library `ABC`:

```
UNLOAD * LIB ABC OBJTYPE N WHERE TRANSFER
```

- Unload all Natural programming objects (source objects only) and user-defined error messages from library `ABC`:

```
UNLOAD * LIB ABC WHERE TRANSFER
```

- Unload all Natural programming objects (source objects only) from library `ABC` with fixed record length:

```
UNLOAD * LIB ABC OBJTYPE N WHERE TRANSFER FIXEDLENGTH
```

## Loading Objects in Internal Format

This section contains command examples of how to load objects from a work file in internal format.

- Load all objects to library `LIBNEW` and replace any that already exist:

```
LOADALL WITH NEWL LIBNEW WHERE REPLACE ALL
```

- Load all object with target library `TGTLIB` to the new target library `NEWTGT`:

```
LOAD * LIB TGTLIB WITH NEWLIBRARY NEWTGT
```

- Load the user-defined error messages `1000` to `1500` from library `ERRLIB` only:

```
LOAD * LIB ERRLIB OBJTYPE E FMNUM 1000 TONUM 1500
```

## Loading Objects in Transfer Format

This section contains command examples of how to load objects from a work file in Transfer format.

- Load all objects to library `LIBNEW` and replace any that already exist:

```
LOADALL WITH NEWL LIBNEW WHERE TRANSFER REPLACE ALL
```

- Load all object with target library `TGTLIB` to new target library `NEWTGT`:

```
LOAD * LIB TGTLIB WITH NEWLIBRARY NEWTGT WHERE TRANSFER
```

# 45 Commands for Navigation and Special Functions

The Object Handler commands in CUI (character user interface) environments are mainly provided for navigation purpose and special function settings such as specifying trace files.

An Object Handler command is entered in the Command line of any Object Handler screen. If you want to execute a Natural system command from an Object Handler screen, enter two slashes (//) before the command. Note that any Natural system command terminates the Object Handler.

### To invoke the Commands menu of the Object Handler

- Choose PF10 (Cmds).

  Or:

  On any Object Handler screen, in the Command line, enter the following:

  ```
  CMDS
  ```

The Object Handler commands are listed below. An underlined portion of a keyword represents an acceptable abbreviation, `Sub` denotes subcommand.

| Command | Sub 1 | Sub 2 | Explanation |
|---|---|---|---|
| CANCEL | | | Cancels the current function and displays the Object Handler **Main Menu**. |
| CHANGE | REPORT | LIBRARY | Invokes the administration function and displays a screen where you can change the report library. |
| CHANGE | WORKPLAN | LIBRARY | Invokes the administration function and displays a screen where you can change the Workplan library. |
| CLEAR | | | Resets the current contents of the input fields in the map to the default values. |
| CMDS | | | Invokes the **Commands** screen. |

| Command | Sub 1 | Sub 2 | Explanation |
|---|---|---|---|
| or<br>COMMANDS | | | |
| BYE<br>EXIT<br>QUIT<br>. | | | Terminates the Object Handler. |
| FIN | | | Terminates the Object Handler and ends the Natural session. |
| GO | HOME | | Displays the Object Handler **Main Menu**. |
| GO | UNLOAD | | Invokes the unload function. |
| GO | UNLOAD | END | Ends the current unload function. |
| | | ERROR | Invokes the unload function for Natural system error messages. |
| | | DDM | Invokes the unload function for DDMs. |
| | | FDT | Invokes the unload function for FDTs. |
| | | LIBRARY | Invokes the unload function for Natural library objects. |
| | | NCP | Invokes the unload function for Natural command processor sources. |
| | | RELATED | Invokes the unload function for Natural-related objects. |
| | | SELECTION<br>or<br>LIST | Displays a screen where you can enter or select the SELECTION or LIST Workplan to be used for the unload function. |
| GO | LOAD | | Invokes the load function. |
| GO | LOAD | ALL | Invokes the load function for all objects contained in the work file. |
| | | END | Ends the current load function. |
| | | ERROR | Invokes the load function for Natural system error messages. |
| | | DDM | Invokes the load function for DDMs. |
| | | FDT | Invokes the load function for FDTs. |
| | | LIBRARY | Invokes the load function for Natural library objects. |
| | | NCP | Invokes the load function for Natural command processor sources. |
| | | RELATED | Invokes the load function for Natural-related objects. |
| | | SELECTION<br>or<br>LIST | Displays a screen where you can enter or select the SELECTION or LIST Workplan to be used for the load function. |
| GO | RESTART | | Displays a screen where you can specify the text object to be used for the restart load function. |

| Command | Sub 1 | Sub 2 | Explanation |
|---|---|---|---|
| GO | SCAN | | Invokes the scan function. |
| GO | SCAN | ALL | Invokes the scan function for all objects contained in the work file. |
| | | END | Ends the current scan function. |
| | | ERROR | Invokes the scan function for Natural system error messages. |
| | | DDM | Invokes the scan function for DDMs. |
| | | FDT | Invokes the scan function for FDTs. |
| | | LIBRARY | Invokes the scan function for Natural library objects. |
| | | NCP | Invokes the scan function for Natural command processor sources. |
| | | RELATED | Invokes the scan function for Natural-related objects. |
| | | SELECTION<br>or<br>LIST | Displays a screen where you can enter or select the SELECTION or LIST Workplan to be used for the scan function. |
| GO | ADMIN | | Invokes the administration function. |
| GO | ADMIN | CHANGE | Displays a screen where you can change the Workplan library. |
| | | CREATE | Opens a menu with which you can create a Workplan. |
| | | LIST | Generates a list of Workplans available in the Workplan library. |
| | | REPORT | Displays a screen where you can change the report library. |
| GO | VIEW | | Invokes the view function. |
| GO | VIEW | DDM | Invokes the view function for DDMs. |
| | | ERROR | Invokes the view function for Natural system error messages. |
| | | FDT | Invokes the view function for FDTs. |
| | | LIBRARY | Invokes the view function for Natural library objects. |
| | | NCP | Invokes the view function for Natural command processor sources. |
| | | RELATED | Invokes the view function for Natural-related objects. |
| GO | FIND | | Invokes the find function. |
| GO | FIND | DDM | Invokes the find function for DDMs. |
| | | ERROR | Invokes the find function for Natural system error messages. |
| | | FDT | Invokes the find function for FDTs. |
| | | LIBRARY | Invokes the find function for Natural library objects. |
| | | NCP | Invokes the find function for Natural command processor sources. |
| | | RELATED | Invokes the find function for Natural-related objects. |
| | | SELECTION<br>or | Displays a screen where you can enter or select the SELECTION or LIST Workplan to be used for the find function. |

| Command | Sub 1 | Sub 2 | Explanation |
|---|---|---|---|
| | | LIST | |
| HELP | | | Invokes the Object Handler help function. |
| INIT | | | Reinitializes the Object Handler utility. |
| READ | PROFILE | | Updates Object Handler settings as defined in the text object PROFILE (see also *Profile Settings*). |
| SET | ADVANCEDCMD | ON | Activates the display of commands generated by the Object Handler in advanced-user and compact input mode. |
| | | OFF | Deactivates the display of commands generated by the Object Handler in advanced-user and compact input mode. |
| | EXECUTIONMSG | ON | Activates a window that displays the processing status. |
| | | OFF | Deactivates a window that displays the processing status. |
| | FREE | ON | Activates free format editing. |
| | | OFF | Deactivates free format editing. |
| | Input-Mode, IM | W | Select **wizard mode**. |
| | | A | Select **advanced user mode**. |
| | | C | Select **compact mode**. |
| | TRACE | ON | Activates trace mode: a trace of each Object Handler action is output to the screen. |
| | | OFF | Deactivates trace mode. |
| | | WORKFILE | Activates trace mode: a trace of each Object Handler action is output to the Natural text object stored in the Workplan library. |
| | TRACEFILE | | Displays a screen where you can specify the Natural text object in the Workplan library to be used for the trace. |
| SETTINGS | | | Displays a screen where you can specify the unload, load or scan settings. |
| SHOW or DISPLAY | LAST | MESSAGE | Displays the last interface return code and message issued by the processing interface of the Object Handler. |
| | | RESULT | Displays the last result issued by the processing interface of the Object Handler. |
| | PROFILE | | Display or modify the Object Handler profile. |
| | REPORT | | Displays the report created last. |
| | STATISTICS | | Displays statistics information about the objects processed. |
| | STATUS | | Displays the current Object Handler status (contents of global variables). |
| | TRACE | FILE | Displays the Natural text object in the Workplan library that contains the trace. |

# 46 Batch Condition Codes and User Exit Routines


- Condition Codes Returned in Batch ..... 278
- Applying User Exit Routines ..... 278
- User Exit Routines Available ..... 279

This section describes the condition codes returned for Object Handler functions in batch mode and the user exit routines available for function processing.

## Condition Codes Returned in Batch

Object Handler processing in batch mode terminates with one of the following condition codes:

| Condition Code | Explanation |
|---|---|
| 0 | Object Handler process terminated successfully. |
| 30 | An internal Object Handler error occurred. |
| 40 | An error was detected in the Object Handler command. |
| 50 | An error occurred during Object Handler processing. |
| 60 | A Natural Security error occurred during Object Handler processing. |
| 99 | A Natural error occurred during Object Handler processing. |

## Applying User Exit Routines

The Object Handler user exit routines are supplied as source objects in the Natural system library SYSOBJH. These source objects are named SRC-EX*nn*, where *nn* denotes the number of the user exit routine.

### To activate a user exit routine

- `CATALOG` or `STOW` source object SRC-EX*nn* under the name `OBJHEX`*nn* in the Natural system library SYSOBJH.

  Different names are used to guarantee that the source object (possibly modified according to your requirements) and the cataloged object of the user exit routine are not overwritten by an update installation.

  For detailed descriptions of the user exit routines, see the source objects of SRC-EX*nn* in the library SYSOBJH.

## User Exit Routines Available

The following user exit routines are available:

- OBJHEX01 for Processing Failures
- OBJHEX02 for Object Rejection
- OBJHEX03 for Default Option Values
- OBJHEX04 for Natural Object Type Statistics

### OBJHEX01 for Processing Failures

Whenever a condition code is set to a value greater than 0 (zero) in batch mode, the user exit routine OBJHEX01 (if available) will be invoked before the Object Handler stops processing. With this user exit routine, you can specify whether to continue or terminate Object Handler processing. In the case of termination, you can change the condition code. For further details, see the source of the user exit routine SRC-EX01 in the Natural system library SYSOBJH.

### OBJHEX02 for Object Rejection

If the Object Handler load function was executed successfully in batch mode (with Condition Code 0) or in online command mode, but one or more objects were rejected during loading (for example, not replaced), before the Object Handler stops processing, the user exit routine OBJHEX02 (if available) is invoked. With OBJHEX02, you can specify whether to continue or terminate Object Handler processing. In the case of termination, you can set a condition code. For further details, see the source of the user exit routine SRC-EX02 in the Natural system library SYSOBJH.

### OBJHEX03 for Default Option Values

You can apply user exit routine OBJHEX03 to set default options for processing Object Handler commands. This user exit is invoked before an Object Handler command is processed. For further details, see the source object of the user exit routine SRC-EX03 in the Natural system library SYSOBJH.

### OBJHEX04 for Natural Object Type Statistics

If the Object Handler unload, load or scan function was executed successfully in command mode, the user exit routine OBJHEX04 (if available) is invoked. It provides statistics for each Natural object type on the sources and catalogued objects processed, replaced, not replaced or rejected. With OBJHEX04, you can specify whether to continue or terminate the Object Handler processing in batch mode. In the case of termination, you can set a condition code. For further details, see the source of the user exit routine SRC-EX04 in the Natural system library SYSOBJH.

# 47 Tools


- Status ..... 282
- Last Result ..... 282
- Traces ..... 282
- Reports ..... 283

The Object Handler provides special features to display status information and reports and to check or modify trace settings.

## Status

Displays the Object Handler functions currently used, the user environment, the Workplan library and the setting of the trace option described below.

### To display the status

- In the Command line of any Object Handler screen, enter the following:

  ```
  SHOW STATUS
  ```

  See also the `SHOW` command described in *Commands for Navigation and Special Functions* in the section *Direct Commands*.

## Last Result

Displays the last internal command issued by the processing interface of the Object Handler and possible return codes and messages.

### To display the last result

- In the Command line of any Object Handler screen, enter the following:

  ```
  SHOW LAST RESULT
  ```

  See also the `SHOW` command described in *Commands for Navigation and Special Functions* in the section *Direct Commands*.

## Traces

Activates or deactivates the trace function. Traces record internal Object Handler program flows to provide control information for error diagnoses. The trace option is set off by default.

### To change the setting

- Use the command `SET TRACE` as described in *Commands for Navigation and Special Functions* in the section *Direct Commands*.

## Reports

Lists the objects loaded, unloaded or scanned, and records errors that may interrupt processing. See also ***Work File Options*** in the section *Settings*. The report option is set on by default and is displayed after the unload, load or scan function has been executed.

### To display the contents of the latest report file

- In the Command line of any Object Handler screen, enter the following:

  ```
  SHOW REPORT
  ```

  See also the `SHOW` command described in *Commands for Navigation and Special Functions* in the section *Direct Commands*.

# 48 Profile Settings


- PF Keys ..... 287
- Line Commands ..... 287
- Profile Parameters ..... 288

Natural provides the option to customize the default settings of your current Object Handler utility environment. For this purpose, Natural provides the text member OBJHPROF in the Natural system library SYSOBJH. OBJHPROF is used to specify environment-specific default values for flags and options that appear when entering the corresponding Object Handler screens.

### To activate individual profile settings

- In online mode, enter the Object Handler internal command `PROFILE` (or `SHOW PROFILE`).

  This command invokes the Profile Maintenance tool that

  - displays a map with the general or user-specific profile parameters and their current values;
  - creates a new Object Handler profile with default values used internally in case an Object Handler profile does not exist;
  - allows you to modify general settings for the profile parameters in the Object Handler profile (controlled by Natural Security);
  - allows you to modify user-specific settings for the profile parameters in the Object Handler profile (controlled by Natural Security);
  - provides a description and help information for each profile parameter.

Enter PF4 to toggle user profile settings. You can use a ***Line Command*** to add, modify, or delete parameters from the user-specific profile.

**Notes:**

1. If new parameters are added, the Profile Maintenance tool internal command `UPDATE` updates the changes into the Object Handler profile. For more information, invoke **Help**.
2. The Object Handler profile itself is named OBJHPROF. It is located in library SYSOBJH. The default profile that is used for updates of the Object Handler profile is named OBJHDEFP. It is also located in library SYSOBJH.

### To deactivate individual profile settings

- Delete the text object OBJHPROF from the library SYSOBJH.

## PF Keys

The following PF keys are available:

| PF key | Description |
|---|---|
| PF1 (**Help**) | Invoke general or context-sensitive help. |
| PF2 (**Print**) | Print the current profile settings. |
| PF3 (**Exit**) | Save the profile data and terminate the Profile Maintenance tool. |
| PF4 (**Gener**/**User**) | Toggle between general (**Gener**) and user-specific (**User**) profile settings. |
| PF5 (**Modif**) | Modify the parameter (marked by cursor selection) in a separate menu. |
| PF6 (**--**) | Scroll to the beginning of the parameter list. |
| PF7 (**-**) | Scroll one page backwards. |
| PF8 (**+**) | Scroll one page forward. |
| PF9 (**++**) | Scroll to the end of the parameter list. |
| PF10 (**ALLDU**/**ALLAU**) | For every parameter: insert line command `DU` or `AU`. |
| PF12 (**Canc**) | Terminate the Profile Maintenance tool. |

## Line Commands

The following line commands are available :

| Line Command | Description |
|---|---|
| MO | Modify parameter value in extended mode. |
| AU | Add entry to user profile |
| DU | Delete entry from user profile |
| DI | Display description of the parameter |
| HE | Display help information on the parameter |

## Profile Parameters

The table below lists the parameters contained in the Object Handler profile OBJHPROF, the possible values that can be entered and the Object Handler functions to which the parameters apply. In addition, the table provides a brief description of the parameters or a reference to the corresponding Object Handler documentation section. Default parameter values are underlined.

| Parameter | Possible Values | Function | Description/ Documentation Section |
|---|---|---|---|
| `Input-Mode` | W | Unload<br>Load<br>Scan | *Wizards* |
| | A | | *Advanced User* |
| | C | | *Compact Mode* |
| `Display-Cmd-in-Advanced-Mode` | N<br>or<br>Y | Unload<br>Load<br>Scan | Displays the Object Handler command generated for a function executed in advanced-user mode. |
| `Display-ExecutionMsg` | N<br>or<br>Y | Unload<br>Load<br>Scan | Activates a window that displays the processing status. |
| `Display-Statistics` | N<br>or<br>Y | Unload<br>Load<br>Scan | Displays statistics on objects processed after the function has been executed. (equivalent to direct command `SHOW STATISTICS`). |
| `Workplan-Library` | WORKPLAN<br>or<br>any other Workplan library | Unload<br>Load<br>Scan<br>Administration | *Workplans* and *Change the Workplan Library* in *Administration* |
| `Workplan-Library-DBID` | 0 (current FNAT/FUSER)<br>or<br>any other Adabas database ID (DBID) | Unload<br>Load<br>Scan<br>Administration | *Change the Workplan Library* |
| `Workplan-Library-FNR` | 0 (current FNAT/FUSER)<br>or<br>any other Adabas file number (FNR) | Unload<br>Load<br>Scan<br>Administration | *Change the Workplan Library* |
| `Report-Library` | WORKPLAN<br>or | Unload<br>Load | `REPORT-LIBRARY` in *Direct Commands, option-setting* |

| Parameter | Possible Values | Function | Description/ Documentation Section |
|---|---|---|---|
| | any other Workplan library | Scan | |
| Report-Library-DBID | 0 (current FNAT/FUSER) or any other Adabas database ID (DBID) | Unload Load Scan | REP-LIB-DBID in *Direct Commands, option-setting* |
| Report-Library-FNR | 0 (current FNAT/FUSER) or any other Adabas file number (FNR) | Unload Load Scan | REP-LIB-FNR in *Direct Commands, option-setting* |
| TRACE | N or Y | Unload Load Scan | ***Traces*** in *Tools* |
| TRACE-TARGET | S (Screen) or W (Work file) | Unload Load Scan | ***Traces*** |
| Option-Replace | N or Y or O (Obsolete) E (Except) | Load | ***Replace Options*** in *Settings* |
| Option-TRANSFER-FORMAT | N or Y | Unload Load Scan | ***Work File Format*** in *Work Files* |
| Option-Use-PC-Work-File | N or Y | Unload Load Scan | See ***Use PC File*** in *Set Additional Options* |
| Option-TR-INCLUDE-LINE-NUMBERS | N or Y | Unload | *Include line numbers* in ***Transfer Options*** (*Settings*) or ***Transfer Options*** (Direct Commands) |
| Option-TR-LINE-NUMBER-INCREMENT | N or Y | Unload Load | **USE-LINE-NUMBER-INCREMENT** in *Transfer Options* (Direct Commands) |
| Option-TR-SUBSTITUTE | N or | Unload | **Substitute line references** in *Transfer Options* (*Settings*) |

| Parameter | Possible Values | Function | Description/ Documentation Section |
|---|---|---|---|
| | Y | | or<br>**SUBSTITUTE** in *Transfer Options* (*Direct Commands*) |
| Option-TR-TRANSLATE-TO-UPPER | N<br>or<br>Y | Load | **Translate to upper case** in *Transfer Options*(*Settings*)<br>or<br>**UPPERCASE-TRANSLATION** in *Transfer Options* (*Direct Commands*) |
| Option-TR-USE-CONVERSION-TABLE | N<br>or<br>S (System table)<br>or<br>U (User table) | Unload<br>Load | **Use conversion table** in *Transfer Options* (*Settings*)<br>or<br>**CONVERSION-TABLE** in *Transfer Options* (*Direct Commands*) |
| Option-TR-CONV-TABLE-NAME-LOAD | OTNCONEA<br>or<br>a user-written subprogram | Load | **Use conversion table** in *Transfer Options* (*Settings*)<br>or<br>**CONVERSION-TABLE** in *Transfer Options* (*Direct Commands*) |
| Option-TR-CONV-TABLE-NAME-UNLD | OTNCONAE<br>or<br>a user-written subprogram | Unload | **Use conversion table** in *Transfer Options* (*Settings*)<br>or<br>**CONVERSION-TABLE** in *Transfer Options* (*Direct Commands*) |
| Option-TR-DA-FORMAT | N<br>or<br>* | Unload<br>Load | **Data area format** in *Transfer OptionsOptions*<br>or<br>**DA-FORMAT** in *Transfer Options* (*Direct Commands*) |
| Option-TR-UNICODE-WORK-FILE | N<br>or<br>Y | UNLOAD | **Unicode work file** in *Settings Screen Fields* |
| Option-TR-LOAD-CODE-PAGE | Code page name<br>or<br>*CODEPAGEY | LOAD | **Use load code page** in *Tansfer Options* (*Settings*) |
| Option-Write-Report | N<br>or<br>Y | Unload<br>Load<br>Scan | ***Reports*** in *Tools*<br><br>**Write report** in *Work File and Report Options* |
| Default-Report-Direct-Command | B<br>or<br>N | Unload<br>Load<br>Scan | Defines the kind of report to be written when a direct command is executed and no report option is |

| Parameter | Possible Values | Function | Description/ Documentation Section |
|---|---|---|---|
| | or<br>Y | | specified in the Object Handler command.<br><br>B Write a batch report. Report data is written directly to the output device. Corresponds to option BATCHREPORT.<br>Y Write a report using a text member. Corresponds to option REPORT.<br>N Do not write a report. Corresponds to option NOREPORT. |
| Default-Report-Option-1 | A<br>or<br>E<br>or<br>S | Unload<br>Load<br>Scan<br>Delete | Defines the option for the report to be written when a direct command is executed and option REPORT-OPTION-1 is not specified in the Object Handler command. For possible values, see REPORT-OPTION-1. |
| Default-Report-Format | S<br>or<br>T | Unload | Defines the format of the report to be written when a direct command is executed and option REPORT-FORMAT is not specified in the Object Handler command. For possible values, see REPORT-FORMAT. |
| Default-Report-Mode | S<br>or<br>L | Unload<br>Load<br>Scan | Defines the report mode to be used when a direct command is executed and option REPORT-MODE is not specified in the Object Handler command. For possible values, see REPORT-MODE. |
| Option-Write-Restart-Info | Y<br>or<br>N | Load | **Write restart information** in *Work File and Report Options* |
| USE-OPTION-WORKPLAN | N<br>or<br>Y | Unload<br>Load<br>Scan | ***Workplans*** |
| OPTION-WORKPLAN-Name | OPTIONWP<br>or<br>any other Workplan of the type OPTION | Unload<br>Load<br>Scan | ***Workplans*** |
| USE-PARAMETER-WORKPLAN | N | Unload | ***Workplans*** |

| Parameter | Possible Values | Function | Description/ Documentation Section |
|---|---|---|---|
| | or<br>Y | Load | |
| PARAMETER-WORKPLAN-Name | PARAWPLN<br>or<br>any other Workplan of the type PARAMETER | Unload<br>Load | *Workplans* |
| WORK-FILE-1-Name | The work file used for the PC file (Work File 7). | Unload<br>Load<br>Scan | *Work Files*<br><br>Only applies if Entire Connection is installed. |
| Report-File-Name | The name of the text member in the Workplan library to be used for the report. | Unload<br>Load<br>Scan | **Write report** in *Work File and Report Options* |
| Restart-File-Name | The name of the text member in the Workplan library to be used for the restart information. | Load | **Write restart information** in *Work File and Report Options* |
| Trace-File-Name | The name of the text member in the Workplan library to be used for the trace. | All functions | *Traces* in *Tools* |
| DELETE-TEMPORARY-REPORT-TEXT | N<br>or<br>Y | Unload Load Scan | Valid on mainframe only: Y = temporary text members for the report are deleted when the Object Handler is terminated (default value) N = temporary text members for the report are not deleted when the Object Handler is terminated |

# 49 Migration from NATUNLD/NATLOAD and SYSTRANS to the Object Handler


- Converting Individual Commands ..... 294
- Processing Commands with a User Exit Routine ..... 296
- Processing SYSTRANS Commands with OBJHAPI ..... 296
- Unsupported SYSTRANS Options ..... 297

You can migrate from the old utilities NATUNLD/NATLOAD and SYSTRANS to the Object Handler by using the two methods described in this section.

## Converting Individual Commands

You can convert NATUNLD/NATLOAD or SYSTRANS direct commands to the corresponding Object Handler commands by using the Object Handler commands provided for migration. These migration commands automatically convert the command syntax used by the old utilities to the command syntax used by the Object Handler.

### To convert a single command

1 Use one of the following Object Handler direct commands:

```
NATUNLD
```

followed by a NATUNLD direct command.

Or:

```
NATLOAD
```

followed by a NATLOAD direct command.

Or:

```
SYSTRANS
```

followed by a SYSTRANS direct command.

The specified utility command is converted to the corresponding Object Handler command.

2 Specify any subsequent command for the Object Handler in the syntax that applies to the utility NATUNLD, NATLOAD or SYSTRANS respectively.

The syntax of this utility remains valid for the duration of the Object Handler session.

**Example of a NATUNLD Command:**

The following is an example of two consecutive NATUNLD utility commands and their corresponding Object Handler commands.

| | |
|---|---|
| Old NATUNLD commands: | `NATUNLD ALL * FM LIB1 TO LIB2` |
| | `ALL PG* FM LIB2` |
| New Object Handler command: | `SYSOBJH NATUNLD ALL * FM LIB1 TO LIB2` |
| Subsequent Object Handler command in NATUNLD syntax: | `ALL PG* FM LIB2` |

**Example of a SYSTRANS Command:**

The following is an example of two consecutive SYSTRANS utility commands and their corresponding Object Handler commands.

| | |
|---|---|
| Old SYSTRANS commands: | `TRANSCMD EXECUTE UNLOAD N FROM LIB1 NAME ETID` |
| | `END` |
| New Object Handler command: | `SYSOBJH SYSTRANS EXECUTE UNLOAD N FROM LIB1 NAME ETID END` |
| Subsequent Object Handler command in SYSTRANS syntax: | `END` |

**Example of SYSTRANS Batch Processing:**

The following is an example of processing a SYSTRANS utility command in batch by using map input data, and the corresponding Object Handler command and input data.

Old SYSTRANS batch sequence:

```
SYSTRANS
U
N,N,N,Y,N,N,N,N
N
SRCLIB1,PGM1,*,TGTLIB1
```

New Object Handler batch sequence:

```
SYSOBJH SYSTRANS
U
N,N,N,Y,N,N,N,N
N
SRCLIB1,PGM1,*,TGTLIB1
```

## Processing Commands with a User Exit Routine

As an alternative to redefining each single utility command, you can specify in a user exit routine that utility commands call `SYSOBJH` for function execution. This routine determines whether or not a direct commands issued to NATUNLD/NATLOD or SYSTRANS is forwarded as an `SYSOBJH` command to the Object Handler.

**To activate the user exit routine**

1. For NATUNLD/NATLOAD:

   `SAVE` source object U-S-EX03 under the name UNLDEX03 in the Natural system library SYSUNLD. The source object is supplied in SYSUNLD.

   For SYSTRANS:

   `SAVE` source object TRA-E2-S under the name TRA-EX-2 in the Natural system library SYS-TRANS. The source object is supplied in SYSTRANS.

2. Open the source of UNLDEX03 or TRA-EX-2 respectively and set `USE-SYSOBJH` to `Y` (this is the default value entered). `N` will deactivate the user exit routine.

   You can define conditions for use such as when the Object Handler is to be used instead of an old utility or who is authorized to use the Object Handler.

3. `CATALOG` or `STOW` the source object.

## Processing SYSTRANS Commands with OBJHAPI

You can use the OBJHAPI Application Programming Interface (supplied in the Natural system library SYSOBJH) to execute an Object Handler command in the syntax of the SYSTRANS utility.

If you use OBJHAPI for this purpose, you have to specify the parameter `P-EXTENSIONS-EXEC-SYSTRANS-CMD` in the program that invokes OBJHAPI. For details, see the example program DOC-API supplied in the library SYSOBJH.

## Unsupported SYSTRANS Options

The Object Handler does not support the following SYSTRANS direct command options: `WORK-FILE-INPUT`, `SPECIAL-CONVERSION`, `RULE-LOAD` and `UNLOAD-RULES`.

# IX Recording Utility

# 50 Recording Utility


- Purpose of Recording ..... 302
- Data and Functions Recorded ..... 302
- Recording a Session ..... 303
- Playing Back a Recording ..... 304
- Manipulating a Recording ..... 306

With the Recording utility, you can record a Natural session and later play back the recorded session.

**Related Documentation:**

*Terminal Commands*

## Purpose of Recording

The Recording utility can be used for the following purposes:

- **Demonstration**
  Instead of having to type in several commands, such as input data by hand, you can play back a recorded sequence of keyboard actions to demonstrate a standard procedure.
- **Application development**
  When applying the same modifications to several objects (for example, programs or maps), you can use a recording to reduce the amount of work involved and at the same time ensure that the modifications are actually the same for all objects affected.
- **Testing**
  You can execute a standard testing procedure by simply playing back a recording.
- **Quality control**
  Before and after making changes to an application, you can play back a recording and compare the results of the two runs to make sure that certain things were not affected by the changes.
- **User training**
  You can incorporate the playback of recordings into training programs for users, to show them specific procedures. Also, you can record user keyboard actions in a session and then inform them of any errors they make or of ways to carry out actions more efficiently. The recording of user actions can also help you to detect any flaws in an application's user interface.

## Data and Functions Recorded

The Recording utility records the following:

- All input data and commands (including terminal commands) entered on the screen.
- Any function keys (PF keys) pressed.
- The current cursor position as contained in the system variable `*CURSOR` (see the *System Variables* documentation).

# Recording a Session

This section describes the steps required to activate and deactivate a recording.


- Specifying Libraries
- Activating a Recording
- Deactivating a Recording


## Specifying Libraries

### To specify the library in which all subsequent recordings are to be stored

- Enter the following terminal command:

```
%B=library-name
```

If you activate the recording process without having specified *library-name*, the name of the library in which the recording is stored is the same as the value of the system variable `*INIT-USER` (see the *System Variables* documentation) at the time when the recording process is activated.

When you log on to another library during a session being recorded, the library in which the recording is being stored remains the same (that is, either the one specified with `%B=` or the `*INIT-USER` library); this means that one recording can record keyboard actions across multiple applications.

## Activating a Recording

### To activate a recording

- Enter the following terminal command:

```
%Bname
```

All subsequent keyboard actions are recorded.

*name* denotes the name under which the data recorded are saved in source form as a Natural object of the type Recording. You can treat this source as any other Natural source (for example, delete it, copy it), except that you must not edit it: recordings contain binary data an editor will destroy.

*name* can only be specified once. If a recording object of the same name already exists in the library specified for recording, Natural returns the message `Error in recording activation`.

**Caution:** Any situation that leads to a backout transaction or rollback (for example, a non-activity timeout) while a recording is in progress, will delete part of the recording thus making the entire recording useless.

Terminal command `%Aname` included in a recording should be followed by terminal command `%B` as described in ***Recording %A***.

### Deactivating a Recording

**To deactivate a recording**

- Enter the following terminal command:

```
%B
```

  The recording has terminated.

## Playing Back a Recording

When a recording is played back, the sequence of, for example, commands and function keys is actually executed again.

The recording is independent of the terminal type; that is, a session recorded on one terminal can be played back on a terminal of another type. You can also play back a recording in batch mode; a recorded online session may, of course, react differently when played back in batch.

This section covers the following topics:

- Step Mode and Background Mode
- Activating a Playback
- Interrupting a Playback

### Step Mode and Background Mode

A recording can be played back in two modes: background mode and step mode.

In background mode, the entire recording is played back invisibly; that is, all keyboard actions of the recording are carried out without anything being displayed to you on the terminal screen during the execution of the recording. You cannot interrupt a recording that is played back in background mode, unless the recording contains the terminal command `%R` as explained in ***Manipulating a Recording***.

In step mode, a recording is played back step by step and all keyboard actions are displayed on the screen. By choosing any function key, you proceed from one step to the next. In step mode, it is also possible for you to interrupt the recording by pressing CLEAR as explained in ***Interrupting a Playback***.

By default, a recording is played back in background mode.

### To set modes

1 To activate step mode, enter the following terminal command:

```
%GON
```

2 To deactivate step mode and activate background mode, enter the following terminal command:

```
%GOFF
```

3 To toggle between step and background mode, enter the following terminal command:

```
%G
```

## Activating a Playback

### To play back a recording

- Enter the following terminal command:

```
%Aname
```

The recording saved under the specified name is executed again.

**Recording** `%Aname`

If you issue the command `%Aname` while a session is being recorded, the recording specified with `%Aname` is not executed but the command `%Aname` is included into the object source that is being recorded. Thus, you can execute a recording from within another recording and concatenate a series of recording to one another. However, you cannot have nested recordings; the execution of the recording that contains the `%Aname` command stops after this command and is not resumed when the execution of *name* finishes. As a result, the data recorded after `%Aname` will never be played back. To avoid this, you should enter `%B` immediately after you have entered `%Aname` in a recording.

## Interrupting a Playback

### To interrupt a recording that is played back in step mode

- Press CLEAR.

Once you have interrupted a recording, you have the following options:

- You can continue your session normally from the point where you stopped the recording.

- You can insert additional keyboard actions into the recording: after you have pressed CLEAR, enter the command %B and all actions you perform are inserted into the source of the recording until you enter %B again. Then, the execution of the recording is resumed.
- You can alter the next step in the recording: after you have pressed CLEAR, enter the command %R, then enter the input data for the next step. The newly entered input data overwrite the input data for this step in the recorded source. When you press ENTER, this step is executed with the new input data and subsequently the execution of the recording is resumed.
- You can execute any helproutine: after you have pressed CLEAR, enter the command %J directly followed by the name of the desired helproutine. The helproutine is invoked and the execution of the recording is continued as soon as the execution of the helproutine ends.

## Manipulating a Recording

By recording the terminal command %R, you can manipulate a single step in a recording when it is played back. This applies in step mode and in background mode. In background mode, %R is the only way to interact with a recording that is being played back. Interaction, for example, may be required to provide an input option for sensitive data, such as passwords which are unknown at the time of the recording.

If the terminal command %R (redisplay last screen) has been recorded, the subsequent screen is open for user input when the recording is played back; that is, the input data for this screen are not taken from the recording but from what the user enters. Subsequently, the execution of the recording is continued.

# X SYSAPI Utility - APIs of Natural Add-On Products

# 51 SYSAPI Utility - APIs of Natural Add-On Products


- Prerequisites ..... 310
- Invoking and Terminating SYSAPI ..... 310
- Reserved Keywords ..... 311
- Using the SYSAPI Utility ..... 311

The utility SYSAPI is used to locate and test Application Programming Interfaces (APIs) provided by Natural add-on products such as Entire Output Management.

The API of a Natural add-on product is a Natural subprogram (cataloged object) that is used for accessing and possibly modifying data or performing services that are specific to an add-on product or a subcomponent.

The API of a Natural add-on product is supplied in the Natural library and/or system file provided for objects that are specific to a particular Natural add-on product. For instructions on using the API of a Natural add-on product, refer to the documentation of the respective add-on product.

For each API of a Natural add-on product, the utility SYSAPI provides a functional description, one example program and API-specific keywords.

**Related Topics:**

- *SYSEXT - Natural Application Programming Interfaces - Utilities* documentation

## Prerequisites

- The appropriate Natural add-on product must be installed at your site.
- The version of the Natural add-on product installed must support the SYSAPI utility features.

## Invoking and Terminating SYSAPI

This section provides instructions for invoking and terminating the SYSAPI utility.

### To invoke the SYSAPI utility

- Enter the following system command:

  ```
  SYSAPI
  ```

  The SYSAPI main menu appears, which lists one or more Natural add-on products, each followed by one or more associated groups of APIs.

  Each group represents a particular API feature provided for the Natural add-on product and contains all example programs that relate to this feature.

  You can select a group to display all belonging APIs in a submenu, see ***Using the SYSAPI Utility***.

**Note:** In the **Command** line, you can enter any Natural system command.

**To terminate SYSAPI**

- Press PF3 or PF12.

## Reserved Keywords

Reserved keywords refer to meta information on APIs, for example the version of a Natural Add-on product in which a new API has been added. Reserved keywords always start with a plus sign (+). See the table below for a description:

| Reserved Keyword | Description |
|---|---|
| +NEW-*PROD-version* | An API that has been added to a specific product in a specific version. |
| +MOD-*PROD-version* | An API that belongs to a specific product and has been modified in a specific version. |

## Using the SYSAPI Utility

After you have invoked the SYSAPI utility, you can select one specific group of APIs by entering any character in the corresponding input field or by cursor selection. The resulting SYSAPI submenu lists all APIs of the group selected in alphabetical order.

This section describes how to use the SYSAPI submenu, for example to submit selection or search criteria. The following topics are covered:

- Elements of the SYSAPI Submenu
- PF Keys
- Line Commands

### Elements of the SYSAPI Submenu

The SYSAPI submenu provides an alphabetical ordered list of APIs. The columns **Interface** and **Description** are headed by selection fields that allow you to enter selection criteria.

The input fields for keywords can be used both for keyword selection and keyword search.

A detailed description of all elements and their usage is given below:

| Element | Explanation | Usage |
|---|---|---|
| **Cmd** | The input field for a line command to be executed on an API, see **Line Commands**. | Enter a line command, example:<br><br>`K` List all keywords relevant to the specified API. |
| **Interface** | The name of the API subprogram. | Enter an asterisk (*) , or a prefix to be delimited by an asterisk. |
| **Description** | A brief description of the purpose of the API. | Enter a string, example:<br><br>`printer` List all APIs with a description containing the string `printer` or `Printer`. |
| **Keywords** | List all keywords or enter a keyword as selection criterion. You can enter at most two keywords. | See ***Keyword Search*** or ***Keyword Selection***. |
| **And/Or** | Specify the logical condition by which multiple keywords are combined: `A` (**And**) or `O` (**Or**). `A` is default. | Enter `A` or `O`. See also ***Keyword Selection***. |
| **Command** | Command line to enter Natural system commands. | Enter a Natural system command. |

**Note:** If a Natural add-on product is localized for some additional language, you can specify this language in your profile parameter settings, see *ULANG - User Language* under *Parameter Reference*. As a result, the list returned for this add-on product contains APIs for the language you have specified.

### Keyword Search

- Enter an asterisk (*), or a prefix delimited by an asterisk, for example:

```
L*
```

As a result, a menu similar to the example below appears.

```
        Search for Keywords

 Mark Keyword
 ---- L*_________________
   _  LIST
   _  LOG
   _  LOGGING
   _  LOGICAL
```

To select a specific keyword as selection criterion, enter any character in the **Mark** column. As a result, a list of all APIs with this selection criterion is displayed.

**Note:** Using keyword search on both input fields results in two menus that are processed and displayed consecutively.

### Keyword Selection

1 Enter a keyword in full. You can also enter keywords in both input fields, for example:

```
USER
```

and

```
LOG
```

2 Choose the logical condition by which multiple keywords are combined: `A` (**And**) and `O` (**Or**).

Depending on the condition specified, you either get a list of all APIs with keyword `USER` **and** keyword `LOG` (And), or a list of all APIs with keyword `USER` **or** keyword `LOG` (Or).

## PF Keys

You can use the following PF keys:

| PF Key | Name | Function |
|---|---|---|
| PF1 | **Help** | Display context-sensitive help. There is a specific help text for each input field. In other contexts, for example the command line, a general help text is displayed. |
| PF2 | **Reset** | Clear all selection fields and readjust the list of APIs. |
| PF3 | **Exit** | Exit the SYSAPI utility, or the current menu or window. |
| PF6 | **--** | Scroll to the beginning of the list. |
| PF7 | **-** | Scroll one page up. |
| PF8 | **+** | Scroll one page down. |
| PF9 | **++** | Scroll to the end of the list. |
| PF12 | **Canc** | Exit the SYSAPI utility, or the current menu or window. |

## Line Commands

Line commands are used to perform object operations. You can enter a line command in the **Cmd** column next to the API required. For a list of valid line commands, enter a question mark (`?`) or press PF1.

The following line commands are available:

| Line Command | Function |
|---|---|
| K | List keywords relevant to the specified API. |
| T | If available, list the text object for a description of the corresponding API. Otherwise, use the line command L to list the example program for a description. |
| L | List the example program. |
| E | Edit the example program. |
| X | Execute the example program. |
| . | Return to the SYSAPI main menu. |

# XI SYSBPM Utility - Buffer Pool Management

The utility SYSBPM is used for managing local and global buffer pools of type `Natural`, `DL/I` or `SORT` and message pools.

In addition to functions for administration, SYSBPM also provides statistical information on the current status of a buffer pool of any type including the buffer pool cache (BP cache) and a message pool (if activated).Information is also provided on the Natural objects loaded in a buffer pool and the BP cache, and the messages loaded in a message pool.

For a general description of the Natural buffer pool (that is the buffer pool of type `Natural`) refer to *Natural Buffer Pool - General*, in the *Operations* documentation.

The buffer pool is defined with the macro `NTBPI` in the Natural parameter module, or with the corresponding dynamic profile parameter `BPI`. Both options are described in the *Parameter Reference* documentation. The type of a buffer pool is determined by the `TYPE` subparameter of `NTBPI` or `BPI`.

**Notes:**

1. Natural objects executed with the `RUN` command will not be loaded into a buffer pool, only Natural objects executed with the `EXECUTE` command, that is the objects must be stowed or cataloged.
2. The BP cache and the message pool are optional storage areas that need to be activated separately.


**Invoking and Operating SYSBPM**

**List Objects**

**Delete Objects**

**Directory Information**

**Hexadecimal Display**

**Write to Work File**

**Display Sorted Extract**

**Buffer Pool Statistics**

**BP Cache Statistics**

**Message Pool Statistics**
**Select Buffer Pool**
**Select Message Pool**
**Blacklist Maintenance**
**Preload List Maintenance**
**Performance Considerations**
**SYSBPM Direct Commands**
**Batch Processing**
**Application Programming Interfaces**

**Note:** In this documentation, the buffer pool is also referred to as “BP”.

# 52 Invoking and Operating SYSBPM


- Invoking and Terminating SYSBPM ..... 318
- Online Help ..... 320
- SYSBPM Main Menu - Fields, Functions and Commands ..... 321
- SYSBPM in a z/OS Parallel Sysplex Environment ..... 324

This chapter describes how to invoke the SYSBPM utility and the usage of the SYSBPM **Main Menu** including the relevant fields, functions and commands. In addition, information is provided on obtaining online help and using SYSBPM in a z/OS Parallel Sysplex environment.

## Invoking and Terminating SYSBPM

### To invoke the SYSBPM utility

- Enter the following Natural system command:

```
SYSBPM
```

The SYSBPM **Main Menu** which looks similar to the example below appears:

```
 16:12:23             ***** NATURAL SYSBPM UTILITY *****            2002-08-27
 BPNAME QA41GBP                 - Main Menu -                  Type Global Nat
 BPPROP OFF                                                      Loc DAEF QA41
                                                               Preload QA41GBPL


            Object Functions                Object Pool Statistics

            L  List Objects                 A  Buffer Pool
            D  Delete Objects               C  BP Cache
            I  Directory Information        M  Message Pool
            H  Hexadecimal Display
            W  Write to Work File           Other Functions
            X  Display Sorted Extract
            ?  Help                         S  Select Buffer Pool
            .  Exit                         B  Blacklist Maintenance
                                            P  Preload List Maintenance
    Code .. _   Library ... *_______
                Object .... *_________________________________________
                DBID ...... 0____  FNR .. 0____  Object Pool ... * (B,C,*) (M)


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

### To display selection fields for message objects

- Enter function code `L` in the `Code..` input field and `M` in the selection field **Object Pool** to display a pop-up menu similar to to the example below:

```
16:12:23              ***** NATURAL SYSBPM UTILITY *****              2002-08-27
BPNAME QA41GBP                  - Main Menu -                    Type Global Nat
BPPROP OFF                                                         Loc DAEF QA41
                                                                 Preload QA41GBPL


         +--------------Specify Message Pool Parameter--------------+
         ! Message number .. 1___ - 9999   Library .. *_______      !
         ! Language code ... 0_            DBID ..... 0____         !
         ! Codepage ........ *_______      FNR ...... 0___          !
         !                                                          !
         +----------------------------------------------------------+
            W  Write to Work File           Other Functions
            X  Display Sorted Extract
            ?  Help                         S  Select Buffer Pool
            .  Exit                         B  Blacklist Maintenance
                                            P  Preload List Maintenance
     Code .. l   Library ... *_______
                 Object .... *_____________________________________
                DBID ...... 0____  FNR .. 0____  Object Pool ... m (B,C,*) (M)


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

### To cancel the pop-up menu

- Press PF3 or PF12.

### To terminate the SYSBPM utility

- Press PF3 or PF12.

  Or:

  In the command line, enter a period (`.`) or enter `EXIT`.

After you have invoked the SYSBPM **Main Menu** you can choose a function code or a PF key from a SYSBPM menu to execute a SYSBPM function. Alternatively, you can use a **SYSBPM direct command** as described in the relevant section.

The functions provided in the SYSBPM **Main Menu** are organized in three sections:

- The **Object Functions** section contains functions for displaying or manipulating Natural objects in the buffer pool or BP cache, and Natural system or user messages in the message pool.
- The **Object Pool Statistics** section contains functions for obtaining object-independent statistical data on the buffer pool or BP cache including hash tables, and the message pool. Object-inde-

pendent data does not include any individual information on the object such as object name, size or addresses.

- The **Other Functions** section contains functions for selecting a buffer pool in the active subsystem (subsid) and for specifying objects to be loaded or not to be loaded into the buffer pool or message pool.

For a description of the available functions, see ***Functions***.

In addition to choosing a function you can also select the Natural objects or messages the function will be applied to by choosing either of the following options:

- Complete the input fields (either for Natural objects or messages) as described in ***SYSBPM Main Menu - Fields, Functions and Commands***.

  Or:
  In the Command line, enter a SYSBPM direct command as described in ***SYSBPM Direct Commands***.

## Online Help

The online help function of SYSBPM provides information on **SYSBPM direct commands** (see the relevant section) or valid input values for fields that appear on SYSBPM screens.

### To invoke the online help function for SYSBPM direct commands

- On any SYSBPM screen, position the cursor in the Command line and press PF1 or enter a question mark (?).

  The **Help** window appears with a list of all SYSBPM direct commands available.

### To invoke the online help function for a SYSBPM input field

- On any SYSBPM screen, position the cursor in any input field and press PF1 or enter a question mark (?).

  The **Help** window appears for the relevant field with a list of all valid input values.

# SYSBPM Main Menu - Fields, Functions and Commands

This section covers the following topics:

- Fields
- Functions
- PF Keys and Direct Commands

## Fields

The following table describes the fields of the SYSBPM **Main Menu**. We also indicate, if a field is specific to only one type of object pool (buffer pool and BP cache vs . message pool).

| Fields | Explanation |
|---|---|
| **BPNAME** | The name of the global buffer pool as specified with the profile parameter `BPNAME`. For a local buffer pool, no name but a blank field is displayed for `BPNAME`.<br><br>See also *BPNAME - Name of Natural Global Buffer Pool* in the *Parameter Reference* documentation. |
| **BPPROP** | The setting of the profile parameter `BPPROP` to control the propagation of changes to an object in a buffer pool.<br><br>See also *BPPROP - Global Buffer Pool Propagation* in the *Parameter Reference* documentation. |
| **Type** | The type of buffer pool, such as `Global Nat`, `Local Nat`, `Global Sort` or `Global DL/I`. |
| **Loc** | The location. Displays the host ID (in the example screen above: `DAEF`) and the subsystem ID (in the example screen above: `QA41`). |
| **Preload** | The name of a preload list if loaded.<br><br>See also ***Preload List Maintenance***. |
| **Library** | The name of the library where the executed object (either from the buffer pool or message pool) is stored.<br>You can specify a name or use asterisk (*) notation.<br><br>The default asterisk (*) selects all libraries.<br><br>This field applies also for message pools. |
| **Object** | The name of the executed object loaded in the buffer pool.<br>You can specify a name or use asterisk (*) notation.<br><br>The default asterisk (*) selects all objects. |
| **DBID** | The database ID (DBID) of the system file FNAT or FUSER where the executed object from the buffer pool or message pool is stored and from where it is loaded.<br><br>If you specify `0` (zero; this is the default) as DBID, the specified object(s) will be selected regardless of their DBID. Any value other than `0` represents a particular DBID specification. |

| Fields | Explanation |
|---|---|
| | This field applies also for message pools. |
| **FNR** | The file number (FNR) of the system file FNAT or FUSER where the executed object from the buffer pool or message pool is stored and from where it is loaded.<br><br>If you specify `0` (zero; this is the default) as FNR, the specified object(s) will be selected regardless of their FNR. Any value other than `0` represents a particular FNR specification.<br><br>This field applies also for message pools. |
| **Object Pool** | Selects the type of object pool(s) to be used.<br><br>`B` — Buffer pool<br>`C` — BP cache.<br>`*` — Both buffer pool and BP cache. This is the default.<br>`M` — Message pool<br><br>**Note:**<br><br>1. If you have selected the message pool (`M`), you can only execute the functions **List Objects**, **Delete Objects**, **Select Buffer Pool** and **Preload List Maintenance**, see *Functions*.<br>2. The type of object pool specified determines the portion of the list of objects that is displayed at first. For example, if you enter selection criterion `C`, the portion contains the objects loaded in the BP cache. By scrolling up, the objects loaded in the buffer pool are displayed. If no object is found in the BP cache, only objects of the buffer pool are displayed.<br>3. If the function **List Objects** is applied to the buffer pool and the BP cache (by selecting `*`), all objects are displayed that are loaded either in the buffer pool or in the BP cache. Objects loaded in the buffer pool are displayed on top. |

### Fields Specific to Messages

The following fields are only available for selecting messages:

| Fields | Explanation |
|---|---|
| **Message Number** | The number of the message. |
| **Language Code** | The language code under which the message has been saved. |
| **Codepage** | The code page under which the message has been saved. |

## Functions

The individual functions are listed below. You invoke a function by entering the one-letter code that corresponds to the function required in the **Code..** field, for example, L for **List Objects**. Note that for the message pool you can only enter the functions **List Objects**, **Delete Objects**, **Select Buffer Pool** and **Preload List Maintenance**.

| Code | Function | Explanation |
|---|---|---|
| L | **List Objects** | Displays information on the objects either loaded in the buffer pool and/or the BP cache (if used), or message pool (if activated). Each list item can be accessed individually and various line commands can be performed for each object. |
| D | **Delete Objects** | Deletes one or more objects from the buffer pool, BP cache or message pool. |
| I | **Directory Information** | Displays the full directory information of a specified object loaded in the buffer pool or the BP cache. |
| H | **Hexadecimal Display** | Displays in hexadecimal format a specified object loaded in the buffer pool. |
| W | **Write to Work File** | Writes to a local file or a PC text file the object directory information located in the buffer pool and/or BP cache. |
| X | **Display Sorted Extract** | Displays a sorted list of 50 object directories located in the buffer pool or BP cache. The list items can be arranged by using any of the sort criteria provided. |
| A | **Buffer Pool** | Invokes the **Buffer Pool Statistics** menu. From this menu, you can invoke object-independent statistics functions for the buffer pool including hash table statistics. |
| C | **BP Cache** | BP cache required.<br><br>Invokes the **BP Cache Statistics** menu. From this menu, you can invoke object-independent statistics functions for the BP cache including hash table statistics. |
| M | **Message Pool Statistics** | Message pool required.<br><br>Invokes the object-independent **Message Pool Statistics** menu. |
| S | **Select Buffer Pool** | Displays a selection list of all available buffer pools. |
| S and specify **Object Pool** M | **Select Message Pool** | Displays a selection list of all available message pools. |
| B | **Blacklist Maintenance** | Invokes the **Blacklist Maintenance** menu which is used to maintain a blacklist of objects which are *not* to be executed. |
| P | **Preload List Maintenance** | Invokes the **Preload List Maintenance** menu for the buffer pool or the message pool (if activated). In a preload list, you can specify the names of objects, that are to be loaded into the buffer pool or the message pool. |

### PF Keys and Direct Commands

In the SYSBPM **Main Menu**, you can use the PF keys or SYSBPM direct commands listed in the table below. An underlined portion of a command represents its minimum abbreviation. For further commands, see ***SYSBPM Direct Commands***.

| PF Key | Command | Function |
|---|---|---|
| PF1 | | Provides SYSBPM help information: see also ***Online Help***. |
| PF3 | EXIT | Leaves the current function/screen and displays the previous screen. |
| PF4 | LAST | Displays the SYSBPM direct command entered most recently. |
| PF6 | FLIP | Switches the PF-key line: toggles between the display of PF1 to PF12 and PF13 to PF24. |
| PF12 | CANCEL | Same as EXIT. |
| PF15 | MENU | Returns to the SYSBPM **Main Menu**. |

## SYSBPM in a z/OS Parallel Sysplex Environment

Whenever Natural switches to another operating system image (host), Natural also switches buffer pools. A switch of buffer pools is indicated by a different host ID, which is displayed in the **Loc** field of a SYSBPM screen.

Switching can take place after each terminal I/O, that is, after choosing any function key or by choosing ENTER. After switching buffer pools, browsing and positioning commands will not be executed (TOP, BOTTOM, +, -, LEFT, RIGHT). Instead, the list starts from the top of the new buffer pool.

If the BPPROP profile parameter (see *BPPROP - Global Buffer Pool Propagation* in the *Parameter Reference* documentation) is set to PLEX or to GPLEX, SYSBPM commands that manipulate blacklists, delete objects or initialize the buffer pool are first executed as usual, and then propagated to other buffer pools available on the same subsystem. If a BP switch caused a function to be aborted or propagated, an appropriate message appears. An appropriate message also appears if Natural has successfully switched to another host and changed buffer pools.

# 53 List Objects


- List Natural Objects ..... 326
- List Messages ..... 338

This function invokes the **List Objects** screen where you can obtain statistical data on the directories of Natural objects currently loaded in the buffer pool or the BP cache (if used). If applied to the message pool, a list of all Natural system or user messages currently loaded in the message pool is displayed.

This chapter provides information on the statistical data displayed on the **List Objects** screen and the commands and functions available for selecting an object or a range of objects, manipulating their current status or navigating in the **List Objects** screen.

Because the **List Objects** screen for Natural objects differs from the screen for messages, each screen is described in a separate chapter:

## List Natural Objects

This section covers the following topics:

- Invoking List Objects for Natural Objects
- List Object Screen for Natural Objects: Columns and Selection Options
- PF Keys and Direct Commands
- Line Commands

### Invoking List Objects for Natural Objects

**To invoke the List Objects screen for Natural objects**

- In the SYSBPM **Main Menu**, enter function code `L` in the **Code ..** input field, and `B`,`C` or `*` in the **Object Pool** selection field and specify the objects, see valid **field input values** described in the section *Invoking and Operating SYSBPM*.

  Or:

  Go directly to the list of objects in the buffer pool by entering the following SYSBPM direct command:

  ```
  DISPLAY LIST library-name object-name dbid fnr
  ```

  Or:

  Go directly to the list of objects in the BP cache by entering the following SYSBPM direct command:

  ```
  DISPLAY CLIST library-name object-name dbid fnr
  ```

A **List Objects** screen similar to the example below appears:

```
 17:13:17              ***** NATURAL SYSBPM UTILITY *****              2002-09-16
 BPNAME QA41GBP                 - List Objects -                  Type Global Nat
 BPPROP OFF                                                         Loc DAEF QA41
 C  Library  Object    DBID  FNR Loc RLD Use Max  Reuse     TotalUC ObjSize Sto
    *_______ *_______ _____ ____ ___ ___ ___ ___ ______ __________ _______ ___
 __ SYSBPM   BPMCALL     10 1640 B         1   1                   5   8,516  12
 __ SYSBPM   BPMNSC      10  410 B             1                   4   3,380   4
 __ SYSDLINP PCNDL02    255  253 B   R         1                  19     292   4
 __ SYSLIBS  NAT00017    10  410 B             1                   1   5,000   8
 __ SYSLIB   ATEST       10 1640 B   R         1                 340  16,148  16
 __ SYSLIB   CATALL10    10  410 B             1                   1   4,256   8
 __ SYSBPM   BPM141-M    10 1640 B             1                   1   5,944   8
 __ SYSDLINS U246005    255  253 B   R         1                  14      52   4
 __ SYSBPM   MENU        10 1640 B             1                   5  10,392  12
 __ SYSLIBS  NAT00040    10  410 B             1                   1   2,816   4
 __ SYSLIBS  NAT00034    10  410 B             1                   1   2,672   4
 __ SYSDLIND DNDL01     255  253 B   R         3                  42     552   4
 __ SYSLIB   ACATALL     10  410 B             1                   3  55,728  56
 __ SYSDLINS U246004    255  253 B   R         2                  28     172   4

Top of List
Command ===>
Enter-PF1---PF2---PF3---PF4---PF5---PF6---PF7---PF8---PF9---PF10--PF11--PF12---
      Help        Exit  Last  Cache --    -     +     ++          >     Canc
```

See also the function **Display Sorted Extract** for a sorted display of objects.

The **List Objects** screen lists all individual objects

1. currently loaded in the buffer pool (first part of the display) and
2. currently loaded in the BP cache (second part of the display).

The statistics displayed are snapshots of the contents of the buffer pool which are refreshed every time you press ENTER.

**Note for GDA Objects Loaded in the Buffer Pool:**

On the **List Objects** screen, two entries may be displayed for a GDA (global data area): one entry contains data in the GDA itself and the other entry contains the internal Natural symbol table for this GDA. This can happen if a program has been cataloged that references a GDA.

## List Object Screen for Natural Objects: Columns and Selection Options

This section describes the columns of the **List Objects** screen for Natural objects, including the associated input fields and selection criteria to shorten the list of objects displayed.

The following topics are covered:

This section covers the following topics:


- Specifying Selection Criteria for the Objects to be Displayed
- Selection Fields for Natural Objects
- Specifying Names, Date and Time


### Specifying Selection Criteria for the Objects to be Displayed

**To specify selection criteria for the objects to be displayed**

- In the input fields underneath the column titles, enter a valid value or range as described for the relevant fields.

  The default value is a blank character or asterisk (*) which selects all objects.

### Selection Fields for Natural Objects

| Column | Explanation |
|---|---|
| **C** | In this column, you can enter a line command to perform a function for the object. See also *Line Commands*. |
| **Library** | The library from which the object was loaded.<br><br>To specify selection criteria, see *Name and Range Specification*. |
| **Object** | The name of the object.<br><br>To specify selection criteria, see *Name and Range Specification*. |
| **DBID** | The database ID of the Natural system file from which the object was loaded.<br><br>To select objects of a specific database, enter a valid numeric value. |
| **FNR** | The file number of the Natural system file from which the object was loaded.<br><br>To select objects of a specific file, enter a valid numeric value. |

| Column | Explanation | |
|---|---|---|
| **Loc** | Location of the object: | |
| | | |
| | B | Buffer pool. |
| | B/C | Buffer pool and BP cache. |
| | C | BP cache. |
| | C/B | BP cache and buffer pool. |
| | | |
| | If B is listed in the first position, the statistical data derive from the buffer pool. If C is listed first, the data derive from the BP cache. Additionally, depending on this positioning, different line commands apply to the fields on the statistics screen (see also *Line Commands*). | |
| | | |
| | To specify the object location(s), enter one of the values below: | |
| | | |
| | B | Selects all objects loaded in the buffer pool only. |
| | B/C | Selects all objects loaded in the buffer pool as well as in the BP cache. |
| | B* | Selects all objects loaded in the buffer pool or in both the buffer pool and BP cache (B* = B + B/C). |
| | C | Selects all objects loaded in the BP cache only. |
| | C/B | Selects all objects loaded in the BP cache as well as in the buffer pool. |
| | C* | Selects all objects loaded in the BP cache or in both the BP cache and buffer pool (C* = C + C/B). |

| Column | Explanation | |
|---|---|---|
| **RLD** | Current status of the object in the buffer pool or the BP cache. A BP cache status only refers to object locking and is therefore only indicated underneath the **L** (Locked) of the **RLD** column. | |
| | | |
| | Buffer pool: | |
| | | |
| | R | Marked as resident. Resident means that the object is not deleted from the buffer pool, not even if the relevant value in the **Use** column changes to 0 (zero) denoting that the object is no longer used. |
| | L | Locked while load function is ongoing. |
| | D | A Delete call for the object is pending. The object will be deleted from the buffer pool as soon as the value in the **Use** column changes to 0 (zero). |
| | | |
| | BP cache: | |
| | | |
| | L | Locked while load function is ongoing. |
| | D | Locked for delete. |
| | | |
| | To select all objects of a specific status, as described above, enter the code R, L or D. | |

| Column | Explanation | |
|---|---|---|
| **Use** | Buffer pool only. | |
| | | |
| | The number of Natural applications that are currently executing the object. | |
| | | |
| | To select objects, you can specify one of the following: | |
| | | |
| | *value* | A numeric value.<br>Selects all objects with this number.<br>Example: 10<br>Selected: 10 |
| | >*value* | A numeric start value (>).<br>Selects all objects with a number greater than *value*.<br>Example: >10<br>Selected: 11, 21<br>Not selected: 10 |
| | <*value* | A numeric end value (<).<br>Selects all objects with a number less than *value*.<br>Example: <10<br>Selected: 9, 8<br>Not selected: 10 |
| **Max** | Buffer pool only.<br><br>The maximum number of applications that have executed the object since it was loaded into the buffer pool.<br><br>To select objects, see the valid input values in **Use** above. | |
| **Reuse** | BP cache only.<br><br>Indicates how many time the object has been loaded (reused) from the BP cache into the buffer pool.<br><br>To select objects, see the valid input values in **Use** above. | |

| Column | Explanation | |
|---|---|---|
| **TotalUC** | Total Use Count: The total number of Locate calls of the object since it was loaded into the buffer pool. | |
| | | |
| | If a BP cache is used, this value is not lost if the object is removed from the buffer pool and saved to the BP cache. Therefore, this value indicates the number of times the object has been used since it was loaded from the system file. | |
| | | |
| | For buffer pool objects, this value is updated regularly. For BP cache objects, this value is only updated after the object was removed from the buffer pool and saved in the BP cache. | |
| | | |
| | To select objects, you can specify one of the following: | |
| | | |
| | *value* or<br>>*value* | A numeric value or a numeric start value (>).<br>Selects all objects with a number greater than or equal to *value*.<br>Example: >10<br>Selected: 10, 11, 21<br>Not selected: 9 |
| | | |
| | <*value* | A numeric end value(<).<br>Selects all objects with a number less than *value* or equal to *value*.<br>Example: <10<br>Selected: 10, 9, 8<br>Not selected: 11 |
| **ObjSize** | The size of the object.<br><br>To select objects, see the valid input values in **TotalUC** above. | |
| **Sto** | Storage that has to be allocated for the object in the buffer pool or BP cache. The text record size of the buffer pool is defined at buffer pool initialization.<br><br>To select objects, see the valid input values in **Use** above. | |
| **BP Load Time*** | The date and time when the object was first loaded into the buffer pool.<br><br>This date and time will be kept until the object has been removed from both the buffer pool and BP cache (deletion from the BP cache only will not remove the display of date and time).<br><br>To select objects, see ***Date Specification*** and ***Time Specification***. | |

| Column | Explanation |
|---|---|
| **BP Last Action*** | Buffer pool only.<br><br>The date and time when the object was last used by an application.<br><br>To select objects, see ***Date Specification*** and ***Time Specification***. |
| **BPC Load Time*** | BP cache (BPC) only.<br><br>The date and time when the object was first loaded into the BP cache.<br><br>This date and time will be kept until the object has been removed from the BP cache.<br><br>To select objects, see ***Date Specification*** and ***Time Specification***. |
| **BPC Last Get*** | BP cache (BPC) only.<br><br>The date and time when the object was last swapped from the buffer pool into the BP cache. This time stamp is also updated if the object was already available in the BP cache and had therefore not been written to the BP cache again.<br><br>To select objects, see ***Date Specification*** and ***Time Specification***. |
| **BPC Last Put*** | BP cache (BPC) only.<br><br>The date and time when the object was last loaded from the BP cache into the buffer pool.<br><br>To select objects, see ***Date Specification*** and ***Time Specification***. |
| **1.BPperiod*** | BP cache (BPC) only.<br><br>The time frame the object has been available in the buffer pool starting with the time the object was first loaded and ending with the time the object was first swapped from the buffer pool into the BP cache. After 24 hours, the display of the time frame is canceled and replaced by this sign: `**:**:**.*` |
| **NatVers*** | The Natural version number an object is cataloged with. |

* press PF11 to display these columns as described in *PF Keys and Direct Commands*

### Specifying Names, Date and Time

This section covers rules on how to specify selection criteria in terms of ranges for the columns Library and Object:

- Name and Range Specification
- Date Specification

- Time Specification

### Name and Range Specification

You can shorten the list of objects displayed on the **List Objects** screen by entering a name or a range of names in the input fields for **Library** and/or **Object**.

In the list of options below, *value* is any combination of one or more characters:

| Input Value | Selected Libraries/Objects |
|---|---|
| * | All libraries/objects.<br><br>This is the default. |
| *value* | All libraries/objects with a name equal to *value*. |
| *value** | All libraries/objects with a name that starts with *value*.<br><br>Example: AB*<br>Selected: AB, AB1, ABC, ABEZ<br>Not selected: AA1, ACB |
| *value*? | All libraries/objects with a name that starts with *value* and ends with any single character for each question mark (?) entered.<br><br>Example: ABC?<br>Selected: ABCA, ABCZ<br>Not selected: AXC, ABCAA |
| *value*?*value*? | All items that match *value* combined with asterisk (*) and question mark (?) in any order.<br><br>Example: A?C*Z<br>Selected: ABCZ, AXCBBBZ, ANCZ<br>Not selected: ACBZ, ABDEZ, AXCBBBZA |
| *value**value*? | |
| **value*?*value** | |
| *value*> | All libraries/objects with a name greater than or equal to *value*.<br><br>Example: AB><br>Selected: AB, AB1, BBB, ZZZZZZZ<br>Not selected: AA1, AAB |
| *value*< | All libraries/objects with a name less than or equal to *value*.<br><br>Example: AX<<br>Selected: AB, AWW, AX<br>Not selected: AXA, AY |

### Date Specification

You can shorten the list of objects displayed on the **List Objects** screen by entering a date, a range of dates, a special date or a range of special dates in the input fields for dates.

A date must be specified in the format *YYYYMMDD* (*YYYY* = year, *MM* = month, *DD* = day).

In the list of options below, *value* is any combination of one or more digits:

| Input Value | Selected Objects |
|---|---|
| *YYYYMMDD* | All objects with a date equal to *YYYYMMDD*.<br><br>Example: 20070630 |
| *value** | All objects with a date that starts with *value*.<br><br>Example: 2007*<br>Selected: 20070101 to 20071231<br>Not selected: 20061231, 20080101 |
| *value*> | All objects with a date greater than or equal to *value*.<br><br>Example: 2007><br>Selected: 20070101 to 20070101<br>Not selected: 20061231 |
| *value*< | All objects with a date less than *value*.<br><br>Example: 2007<<br>Selected: 20060101 to 20061231<br>Not selected: 20070101, 20071231 |
| **Special Dates** | |
| TODAY<br><br>or<br>TO+/-*n* | All objects with the date of the current day or a day before or after the current day:<br><br>TODAY — All objects with the date of the current day.<br><br>TO+/-*n* — All objects with the date of the current day plus or minus *n* days.<br><br>Example: The current date is June 30th of 2007.<br>TO-5 selects 20070625. |
| YESTERDAY | All objects with the date of the day before the current day.<br><br>Example: The current date is June 30th of 2007.<br>YESTERDAY selects 20070629. |
| MONTH | All objects with the date range of the current month.<br><br>Example: The current month is June of 2007.<br>MONTH selects 20070601 to 20070630. |

| Input Value | Selected Objects |
|---|---|
| YEAR | All objects with the date range of the current year.<br><br>Example: The current year is 2007.<br>YEAR selects 20070101 to 20071231. |

### Time Specification

You can shorten the list of objects displayed on the **List Objects** screen by entering a time or a range of times in the input fields for times.

The time must be specified in the format *HH*:*II*:*SS* (*HH* = hours, *II* = minutes, *SS* = seconds).

In the list of options below, *value* can be any combination of one or more digits:

| Input Value | Selected Objects |
|---|---|
| *HH*:*II*:*SS* | All objects with a time equal to *HH*:*II*:*SS*.<br><br>Example: 14:15:16 |
| *value** | All objects with a time that starts with *value*.<br><br>Example: 13*<br>Selected: 13:00:00, 13:10:53, 13:59:59<br>Not selected: 12:59:59, 14:00:00 |
| *value*> | All objects with a time greater than or equal to *value*.<br><br>Example: 12:30><br>Selected: 12:30:00, 12:30:01, 16:34:01<br>Not selected: 12:29:59 |
| *value*< | All objects with a time less than *value*.<br><br>Example: 12:30<<br>Selected: 12:29:59<br>Not selected: 12:30:00 |

## PF Keys and Direct Commands

On the **List Objects** screen, you can use the PF keys or SYSBPM direct commands listed in the table below. An underlined portion of a command represents its minimum abbreviation. For further commands, see ***SYSBPM Direct Commands***.

| PF Key | Command | Function |
|---|---|---|
| PF1 | | Provides SYSBPM help information.<br>If chosen with the cursor on column **C**:<br>lists all commands and functions available.<br><br>If chosen with the cursor on the input fields underneath the column titles: lists all possible input values for object selection. |
| PF3 | EXIT | Leaves the current function/screen and displays the previous screen. |
| PF4 | LAST | Displays the SYSBPM direct command entered most recently. |
| PF5 | CACHE | Only applicable if BP cache data exists.<br><br>Scrolls to the top of the list with statistical data on BP cache objects. |
| PF6 | - | Scrolls to the top of the list with statistical data on buffer pool objects. |
| PF7 | - | Scrolls one page up in a list. |
| PF8 | + | Scrolls one page down in a list. |
| PF9 | ++ | Scrolls to the end of the list. |
| PF10 | < | Scrolls left in the list. |
| | LEFT | Press PF11 to scroll to the right. |
| PF11 | > | Scrolls right in the list and displays the additional screen columns: **BP Load Time**, **BP Last Action**, **BPC Load Time**, **BPC Last Get**, **BPC Last Put** and **1.BPperiod**. |
| | RIGHT | Press PF10 to scroll to the left. |
| PF12 | CANCEL | Same as EXIT. |
| PF15 | MENU | Return to the SYSBPM **Main Menu**. |

## Line Commands

On the **List Objects** screen, in column **C**, for each object displayed, you can enter one of the line commands listed below:

| Command | Function |
|---|---|
| CL | Buffer pool only.<br><br>Releases an object marked as resident. |
| DE | Marks an object to be deleted from the buffer pool or BP cache.<br><br>The object is deleted as soon as the relevant **Use** count changes to 0 (object no longer used). If issued for a buffer pool object, the object will be deleted from both the buffer pool and the BP cache. If issued for a BP cache object, the object will be deleted from the BP cache only. |
| HD | Buffer pool only.<br><br>Displays in hexadecimal format the directory information of an object. |

| Command | Function |
|---|---|
| HE | Buffer pool only.<br><br>Corresponds to the function **Hexadecimal Display** as described in the relevant section. |
| FO | Buffer pool only.<br><br>Deletes an object immediately from the buffer pool, regardless of the relevant **Use** count. |
| LD | Corresponds to the function **Directory Information** as described in the relevant section. |
| RE | Buffer pool only.<br><br>Marks an object as resident. |
| ZO | Zooms in the fields **Object**, **Use**, **Max**, **Reuse**, **TotalUC**, **ObjSize** and **Sto** and displays them in full length.<br>To zoom out, press ENTER. |

For each command entered, a confirmation message appears for the relevant line overwriting text of rows displayed on the screen. Possible messages are:

- `Failed` (in response to any function that has not been executed successfully),
- `Deleted` (in response to the command `DE` or `FO`),
- `Released` (in response to the command `CL`) and
- `Resident` (in response to the command `RE`).

## List Messages

This section covers the following topics:


- Invoking List Objects for Natural System and User Messages
- List Object Screen for Messages: Columns and Selection Options
- PF Keys and Direct Commands
- Line Commands


### Invoking List Objects for Natural System and User Messages

**To invoke the List Objects screen for system and user messages**

- In the SYSBPM **Main Menu**, enter function code `L` in the **Code..** field and selection criterion `M` in the **Object Pool** selection field and specify the messages, see valid **field input values** described in the section *Invoking and Operating SYSBPM*.

  Or:

Go directly to the list of objects in the buffer pool by entering the following SYSBPM direct command:

```
DISPLAY MLIST start-number end-number library-name language dbid fnr codepage
```

A **List Objects** screen similar to the example below appears:

```
 17:13:17              ***** NATURAL SYSBPM UTILITY *****             2002-09-16
 BPNAME QA41GBP                - List Objects -                 Type Global Nat
 BPPROP OFF                                                       Loc DAEF QA41
  C  Library  Nmbr  DBID  FNR Code Page LC Message Text
     *_______ ____ _____ ____ *_______  __
  __ XXXLIB     25   177    8 CPAGE02    2 U0025 TEST TEST TEST
  __ <NATSYS>   80    10 2430 IBM01141   1 Command / program name must start wi
  __ <NATSYS>   82    10 2430 IBM01141   1 Invalid command, or :1: :2: does not
  __ XXXLIB    102   177    8 CPAGE02    2 U0102 TEST TEST TEST
  __ GGSLIB    296   177    7 CPAGE22   34 U0296 TEST TEST TEST
  __ <NATSYS>  631    10 2430 IBM01141   1 Invalid index specified in arithmeti
  __ <NATSYS>  660    10 2430 IBM01141   1 Time-stamp inconsistency for segment
  __ XXXLIB    759   177    8 CPAGE02    2 U0759 TEST TEST TEST
  __ XXXLIB    789   177    8 CPAGE02    2 U0789 TEST TEST TEST
  __ XXXLIB    806   177    8 CPAGE02    2 U0806 TEST TEST TEST
  __ XXXLIB    853   177    8 CPAGE02    2 U0853 TEST TEST TEST
  __ <NATSYS>  932    10 2430 IBM01141   1 Program version error.
  __ <NATSYS>  933    10 2430 IBM01141   1 GDA time-stamp conflict.
  __ <NATSYS>  933    10 2430 IBM01140   1 GDA time-stamp conflict.
Top of List
Command ===>
Enter-PF1---PF2---PF3---PF4---PF5---PF6---PF7---PF8---PF9---PF10--PF11--PF12---
      Help        Exit              --    -     +     ++                Canc   ↩
```

The **List Objects** screen lists all individual messages currently loaded in the message pool. It will be refreshed every time you press `ENTER`.

### List Object Screen for Messages: Columns and Selection Options

This section describes the columns of the **List Objects** screen for messages, including the associated input fields and selection criteria to shorten the list of objects displayed.

- Specifying Selection Criteria for the Objects to be Displayed
- Selection Fields for Messages

- Name and Range Specification

### Specifying Selection Criteria for the Objects to be Displayed

**To specify selection criteria for the objects to be displayed**

- In the input fields underneath the column titles, enter a valid value or range as described for the relevant fields.

  The default value is a blank character or asterisk (*) which selects all objects.

### Selection Fields for Messages

This section gives a tabular overview on Fields of the **List Object** menue.

| Column | Explanation |
|---|---|
| **C** | In this column, you can enter a line command to perform a function for the corresponding message. See also *Line Commands*. |
| **Library** | The library from which the message was loaded.<br><br>To specify selection criteria, see *Name and Range Specification*. |
| **Nmbr** | The number of the message.<br><br>To select messages of a specific number, enter a valid numeric value. |
| **DBID** | The database ID of the Natural system file from which the message was loaded.<br><br>To select a message of a specific database, enter a valid numeric value. |
| **FNR** | The file number of the Natural system file from which the message was loaded.<br><br>To select messages of a specific file, enter a valid numeric value. |
| **Code Page** | The code page under which the message has been saved. |
| **LC** | The language code under which the message has been saved. |
| **Message Text** | The beginning of a message short text. |

* press PF11 to display these columns as described in *PF Keys and Direct Commands*

### Name and Range Specification

See the *Name and Range Specification* section in the *List Natural Objects* chapter.

## PF Keys and Direct Commands

On the **List Objects** screen, you can use the PF keys or SYSBPM direct commands listed in the table below. An underlined portion of a command represents its minimum abbreviation. For further commands, see *SYSBPM Direct Commands*.

| PF Key | Command | Function |
|---|---|---|
| PF1 | | Provides SYSBPM help information.<br>If chosen with the cursor on column **C**:<br>lists all commands and functions available.<br><br>If chosen with the cursor on the input fields underneath the column titles: lists all possible input values for object selection. |
| PF3 | EXIT | Leaves the current function/screen and displays the previous screen. |
| PF6 | - | Navigates to the top of the list of messages in the message pool. |
| PF7 | - | Scrolls one page up in a list. |
| PF8 | + | Scrolls one page down in the list. |
| PF9 | ++ | Navigates to the end of the list. |
| PF12 | CANCEL | Same as EXIT. |
| PF15 | MENU | Returns to the SYSBPM **Main Menu**. |

## Line Commands

On the **List Objects** screen, in column **C**, for each object displayed, you can enter one of the line commands listed below:

| Command | Function |
|---|---|
| DE | Deletes a message from the message pool. The entry is replaced by the confirmation “deleted”. |

# 54 Delete Objects


- Delete Objects from Buffer Pool and/or BP Cache ..... 344
- Delete Objects from Message Pool ..... 344

This function is used to delete one or more objects from the buffer pool (BP) and/or the BP cache. You can also apply the function to the message pool to select messages to be deleted.

## Delete Objects from Buffer Pool and/or BP Cache

An object that has a **Current Use Count** (see *Directory Information*) of `0` (zero) is deleted immediately. `0` denotes that such an object is no longer used. An object with a **Current Use Count** greater than `0` is marked for deletion and deleted as soon as its **Current Use Count** changes to `0`.

### To delete Natural objects

- In the SYSBPM **Main Menu**, enter function code `D`, the code for the type of object pool (`B`, `C` or `*`), and specify the object(s) to be deleted, see valid **field input values** in *Invoking and Operating SYSBPM*for valid values.

  Or:

  Enter the following SYSBPM direct command:

  ```
  DELETE library-name object-name dbid fnr
  ```

## Delete Objects from Message Pool

### To delete messages

- In the SYSBPM **Main Menu**, enter function code `D`, the code for the message pool (`M`), and specify the message object(s) to be deleted, see the valid **field input values** in *Invoking and Operating SYSBPM*.

  Or:

  Enter the following SYSBPM direct command:

  ```
  DELETE MP start-number end-number library-name language dbid fnr codepage
  ```

# 55 Directory Information


- Fields for Buffer Pool Objects ..... 346
- Fields for BP Cache Objects ..... 348
- PF Keys and Direct Commands ..... 349

This function is used to display the full directory of a Natural object currently loaded in the buffer pool or BP cache.

### To invoke Directory Information

- In the SYSBPM **Main Menu**, specify the following:

  Enter function code `I`.

  In the fields **Object**, **Library**, **DBID** and **FNR**, specify the object for which to display the directory: the valid **input values** are described in *Invoking and Operating SYSBPM*.

  In the **Object Pool** field, enter the code that corresponds to the object pool from which to read the object directory information:

| | |
|---|---|
| `B` or * (asterisk) | Buffer pool |
| `C` | BP cache |

  Or:

  Use either of the following SYSBPM direct commands:

  - `DISPLAY DIRECTORY library-name object-name dbid fnr`

    (For the directory of the specified object loaded in the buffer pool.)
  - `DISPLAY CDIRECTORY library-name object-name dbid fnr`

    (For the directory of the specified object loaded in the BP cache.)

  The **Directory Information** screen appears. Depending on the object pool (buffer pool or BP cache) selected, the screen provides different fields as described in the following section.

## Fields for Buffer Pool Objects

For directories of objects loaded in the buffer pool, the **Directory Information** screen provides the following fields and information on a specified object:

| Field | | Explanation |
|---|---|---|
| **Directory of** | | The type (for example, map) and name of the object. |
| **Loaded** | **from Library** | The name of the library from which the object was loaded into the buffer pool. |
| | **on DBID/FNR** | The database ID (DBID) and file number (FNR) of the system file FNAT or FUSER from which the object was loaded into the buffer pool. |
| | **on** | The date and time when the object was loaded into the buffer pool. |
| | **by User** | The ID of the user who executed the object. |
| **Last Action on** | | The date and time when the object was last used by an application. |
| **BP Directory at Address** | | The address of the directory of the object in the buffer pool. |
| **Object at Address** | | The address of the object in the buffer pool. |
| **Allocated Size (KB)** | | The size that has been allocated in the buffer pool for the object. |
| **Object Size** | | The size of the object. |
| **Status (RLD)** | | The status of the object:<br>`R` — The object is resident in the buffer pool. Resident means that the object is not deleted from the buffer pool, not even if its **Current Use Count** (see below) changes to `0` (zero).<br>`L` — The object is currently locked.<br>`D` — A Delete call for the object is pending. The object will be deleted from the buffer pool as soon as its **Current Use Count** (see below) changes to `0` (zero). |
| **Current Use Count** | | The number of applications currently executing the object.<br><br>A value of `0` (zero) denotes that the object is no longer used. |
| **Maximum Use Count** | | The maximum number of applications that have executed the object since it was loaded into the buffer pool. |
| **BP Total Use** | | The total number of times an object has been executed since it was loaded from the system file into the buffer pool.<br><br>If a BP cache is used, this value is not lost if the object is removed from the buffer pool and saved to the BP cache. Therefore, this value indicates the number of times the object has been used since it was loaded from the system file. |
| **Cataloged** | | The information displayed in the **Cataloged** section of the **Directory Information** screen is identical to the information provided with the Natural system command `LIST DIRECTORY` described in the *System Commands* documentation. |

## Fields for BP Cache Objects

For directories of objects loaded in the BP cache, the **Directory Information** screen provides the following fields and information on a specified object:

| Field | Explanation |
|---|---|
| *object-type* | The type (for example, map) and name of the object. |
| **Library** | The name of the library from which the object was loaded into the buffer pool. |
| **DBID** | The database ID of the system file FNAT or FUSER from which the object was initially loaded into the buffer pool. |
| **FNR** | The file number of the system file FNAT or FUSER from which the object was initially loaded into the buffer pool. |
| **Last Put** | The date and time when the object was last loaded from the BP cache into the buffer pool. |
| **BP Load Time** | The date and time when the object was first loaded into the buffer pool. |
| **Cache Load Time** | The date and time when the object was first loaded into the BP cache. |
| **Last Get** | The date and time when the object was last swapped from the buffer pool into the BP cache. |
| **Position Index** | Serially numbered internal Natural position index of the objects in the BP cache. |
| **First Data Block Offset** | The address of the directory of the object in the BP cache. |
| **Allocated Size (KB)** | The size that has been allocated in the BP cache for the object. |
| **Object Size** | The size of the object. |
| **Status** | The status of the object:<br>`L` Locked while load function is ongoing.<br>`D` Locked for delete. |
| **Reuse** | Indicates how many times the object has been returned from the BP cache to the buffer pool. |
| **BP Total Use** | The total number of times an object has been executed since it was initially loaded from the system file into the buffer pool and then into the BP cache. |
| **Cataloged** | The information displayed in the **Cataloged** section of the **Directory Information** screen is identical to the information provided with the Natural system command `LIST DIRECTORY` described in the *System Commands* documentation. |

## PF Keys and Direct Commands

On the **Directory Information** screen, you can use the PF keys or SYSBPM direct commands listed in the table below. An underlined portion of a command represents its minimum abbreviation. For further commands, see *SYSBPM Direct Commands*.

| PF Key | Command | Function |
|---|---|---|
| PF1 | | Provides help information on SYSBPM direct commands. |
| PF2 | NEXT | Only applies if a range of objects was selected.<br>Displays one object after the other and then redisplays the screen on which NEXT was entered. |
| PF3 | EXIT | Leaves the current function/screen and displays the previous screen. |
| PF4 | LAST | Displays the SYSBPM direct command entered most recently. |
| PF6 | FLIP | Switches the PF-key line: toggles between the display of PF1 to PF12 and PF13 to PF24. |
| PF12 | CANCEL | Same as EXIT. |
| PF15 | MENU | Invokes the SYSBPM **Main Menu**. |
| | FDELETE | Buffer pool only.<br><br>Deletes an object immediately from the buffer pool, regardless of its **Use** count. |
| | RESIDENT | Buffer pool only.<br><br>Marks an object as resident. Resident means that the object is not deleted from the buffer pool, not even if its **Use** count is 0 (object no longer used). |
| | CLEAR | Buffer pool only.<br><br>Releases an object marked as resident. |
| | DELETE | Marks an object for deletion. See Status D of the **buffer pool** or **BP cache** mentioned in a previous section. |

# 56 Hexadecimal Display


- PF Keys and Direct Commands ..... 352

This function is used to display in hexadecimal format the code of a Natural object currently loaded in the buffer pool.

### To invoke Hexadecimal Display

- In the SYSBPM **Main Menu**, enter function code `H` and specify an object: see the valid **field input values** as described in the section *Invoking and Operating SYSBPM*.

  Or:

  Enter the following SYSPBM direct command:

  ```
  DISPLAY HEX library-name object-name dbid fnr
  ```

  The **Hexadecimal Display** screen appears with the object code displayed in hexadecimal format.

## PF Keys and Direct Commands

Within the object displayed on the screen, you can move to a specific location by entering either an absolute hexadecimal address or a hexadecimal offset relative to your current position.

On the **Hexadecimal Display** screen, you can use the PF keys or SYSBPM direct commands listed in the table below. An underlined portion of a command represents its minimum abbreviation. For further commands, see ***SYSBPM Direct Commands***.

| PF Key | Command | Function |
|---|---|---|
| PF1 | | Provides help information on SYSBPM direct commands. |
| PF2 | `NEXT` | Only applies if a range of objects was selected.<br>Displays one object after the other and then redisplays the screen on which `NEXT` was entered. |
| PF3 | `EXIT` | Leaves the current function/screen and displays the previous screen. |
| PF4 | `LAST` | Displays the SYSBPM direct command entered most recently. |
| PF6 | `-` | Scrolls to the top of the display. |
| PF7 | `-` | Scrolls up one page. |
| PF8 | `+` | Scrolls down one page. |
| PF9 | `++` | Scrolls to the end of the display. |
| PF12 | `CANCEL` | Same as `EXIT`. |
| PF15 | `MENU` | Invokes the SYSBPM **Main Menu**. |
| | `NEXT` | Only applies if a range of objects was selected.<br>Displays one object after the other and then redisplays the screen on which `NEXT` was entered. |

# 57 Write to Work File

This function writes to a work file the directory information of Natural objects currently loaded in the buffer pool and/or BP cache.

### To invoke Write to Work File

1 In the SYSBPM **Main Menu**, enter the following:

- In the **Code** field, enter function code `W`.
- In the **Object Pool** field, enter the code that corresponds to the object pool from which to read the object directory information:

| | |
|---|---|
| `B` | Buffer pool. |
| `C` | BP cache. |
| * (asterisk) | Both buffer pool and BP cache. |

Or:

Enter one of the following SYSBPM direct commands:

- `WRITE ALL`

  (saves buffer pool and BP cache data)
- `WRITE BP`

  (saves buffer pool data only)
- `WRITE BPC`

  (saves buffer pool data only)

The **Work File Selection** window appears.

2 Specify the following:

- Select the target work file:

  Enter `N` (No; this is the default setting) to output the data on Natural Work File 1.

  Or:
  If Entire Connection is installed, enter `Y` (Yes) to output the data on a PC text file by using Natural Work File 7.

- Enter the delimiter character, for example a semi-colon (;), to be used for separating the columns in the work file. The default is a blank character.

The statistical data written to the work file are snapshots of the list generated by the **List Objects** function. Refer to ***List Objects*** for explanations of the columns.

The PC text file can used as the basis for spreadsheet calculation.

# 58 Display Sorted Extract

This function generates a sorted excerpt of 50 directory entries of Natural objects currently loaded in the buffer pool. This list can be used for evaluation purposes, such as determining the objects to be marked as resident or to be included in a preload list as described in the relevant section.

### To invoke Display Sorted Extract

- In the SYSBPM **Main Menu**, enter function code `X` and, in the **Object Pool** field, specify the type of pool by choosing `B` (buffer pool) or `C` (BP cache).

  The **Specify Sort Criteria** window appears. In the input fields, enter any of the codes below to choose a column and the order by which to sort the statistics columns:

| | | |
|---|---|---|
| **Sort Value** | `O` | **ObjSize** = Object Size. This is the default. |
| | `T` | **TotalUC** = Total Use Count. See the description of **BP Total Use** in *Directory Information*. |
| | `L` | **BP Last Action** (only applicable to the buffer pool). The date and time when the objects were last used by an application. |
| **Sort Order** | `D` | Descending order. This is the default. |
| | `A` | Ascending order. |

  Or:

  Use the SYSBPM direct command `SORT` or `SORT BPC` as described in the section *SYSBPM Direct Commands*.

  The **BP Extract** screen appears which indicates the sort criteria specified.

The **BP Extract** screen is similar to the **List Objects** screen. For explanations of the columns and the commands that apply, refer to ***List Objects***.

# 59 Buffer Pool Statistics


- General Buffer Pool Statistics ..... 358
- Buffer Pool Load/Locate Statistics ..... 361
- Buffer Pool Fragmentation ..... 365
- Internal Function Usage ..... 367
- Buffer Pool Hash Table Statistics ..... 367
- Performance Hints ..... 373
- PF Keys and Direct Commands ..... 378

The **SYSBPM** function invokes the **Buffer Pool Statistics** menu, which is used to obtain buffer-pool-related statistics (including hash table statistics).

### To invoke Buffer Pools Statistics

- In the SYSBPM **Main Menu**, enter the following function code:

  ```
  A
  ```

  Or:

  Enter the following SYSBPM direct command:

  ```
  DISPLAY STATISTICS
  ```

  The **Buffer Pool Statistics** menu appears.

The functions available in the **Buffer Pool Statistics** menu and the commands provided on the screens that are invoked by these functions are described in this section.

## General Buffer Pool Statistics

This function is used to monitor the performance of the buffer pool, and displays statistics regarding the activity of the buffer pool.

### To invoke General Buffer Pool Statistics

- In the **Buffer Pool Statistics** menu, enter the following function code:

  ```
  G
  ```

  Or:

  Enter the following SYSBPM direct command:

  ```
  DISPLAY GENERAL
  ```

  The **General Buffer Pool Statistics** screen appears.

The statistics displayed on the **General Buffer Pool Statistics** screen are snapshots of the buffer pool which are refreshed each time you press ENTER. The following information is displayed:

| Field | Explanation |
|---|---|
| **Buffer Pool Address** | The address of the buffer pool. |
| **Directory Section** | The address of the buffer pool directory section relative to the beginning of the buffer pool.<br><br>Each object loaded in the buffer pool requires a directory entry that contains information on this object. The space for these directory entries is allocated in the buffer pool. |
| **Text Record Section** | The address of the text record section relative to the beginning of the buffer pool.<br><br>After the space used by the directory entries has been allocated, the remaining space is divided into blocks called text records (whose size, by default, is 4 KB). An object can occupy one or more text records, depending on its size. |
| **Dataspace attached** | The name of the dataspace (BP cache) attached to the buffer pool. |
| **Buffer Pool Size (MB)** | The size of the whole buffer pool in MB.<br><br>The buffer pool size can be specified with the NTBPI macro in the parameter module or with the `BPI` profile parameter described in the *Parameter Reference* documentation. |
| **Directory Entry Size** | The size of a directory entry in bytes. |
| **Text Record Size (KB)** | The size of a text record in KB. The text record size can be specified with the NTBPI macro in the parameter module or with the `BPI` profile parameter described in the *Parameter Reference* documentation.<br><br>You can change the text record size of an existing buffer pool if you reinitialize the buffer pool by using the `INITIALIZE` command.<br><br>The default text record size is set to 4 KB. However, if you use applications that consist of many rather small objects, we recommend that you reduce it to 2 KB. This reduces the percentage of unused space in the buffer pool, although it can lead to Algorithm 2 (see *METHOD=S* in the *Operations* documentation) being invoked more frequently. |
| **Buffer Pool Start** | The date and time when the buffer pool was originally started. |
| **Last Initialization** | The date and time when the buffer pool was most recently initialized, and the ID of the user who performed the initialization.<br><br>The buffer pool is initialized when:<br><br>■ originally starting the buffer pool,<br>■ executing the `INITIALIZE` SYSBPM direct command, or<br>■ executing the `REFRESH` command of the GBP operating program; see also *Global Buffer Pool Operating Functions* in the *Operations* documentation. |
| **Text Records - Total** | The total number of text records. |

| Field | Explanation |
|---|---|
| **Text Records - Used** | The number of text records currently used. |
| **Text Records - Used in %** | The percentage of text records currently used. |
| **Text Records - Max Used** | The maximum number of text records used. |
| **Text Records - Total Size** | The total space used by all text records used, which is **Text Records - Used** multiplied by the size of a single text record.<br><br>The difference between the total text record size and the total object size shows the amount of unused size in the text record section and can also be an indicator for the system administrator of whether to modify the text record size. |
| **Text Records - Avg Usage %** | The average usage in percent of all text records used, which is **Objects - Total Size** divided by **Text Records - Total Size**.<br><br>This value should not be significantly less than 75%. If the buffer pool is almost full, any value above 75% indicates good usage of the buffer pool. If the usage is significantly less than 75%, the text record size should be reduced. |
| **Space Used %** | The actual usage in percent of the text record section, which is **Objects - Total Size** divided by the total size of the **Text Records** section.<br><br>**Tip:** If the buffer pool is almost full (that is, the value in the field **Text Records - Used** is almost 100%), any value above 75% indicates good usage of the buffer pool. If the usage is significantly less than 75%, the text record size should be reduced. |
| **Objects - Loaded** | The number of objects currently loaded in the buffer pool. |
| **Objects - Max Loaded** | The maximum number of objects ever loaded simultaneously in the buffer pool since the buffer pool was started. |
| **Objects - Total Size** | The total size in bytes of the objects currently loaded. |
| **Objects - Avg TR Used** | The average number of text records used by one object. |
| **Objects - SumOfUseCounts** | Totals the use counts of all objects currently loaded in the buffer pool.<br><br>The use count counts all applications currently executing an object. If an object is currently not in use, its use count changes to 0 (zero). |
| **Objects - AvgLifetimeUsed(min)** | The average life time (in minutes) of objects currently loaded in the buffer pool. |
| **Objects - AvgLifetimeReplace(min)** | The average life time (in minutes) of objects, which have already been replaced in the buffer pool. |

# Buffer Pool Load/Locate Statistics

This function provides statistical information on the loading of objects into the buffer pool and the locating of objects in the buffer pool. This information also serves as an indicator of buffer pool performance.

### To invoke Buffer Pool Load/Locate Statistics

- In the **Buffer Pool Statistics** menu, enter the following function code:

  ```
  L
  ```

  Or:

  Enter the following SYSBPM direct command:

  ```
  DISPLAY LOAD
  ```

  The **Buffer Pool Load/Locate Statistics** screen appears.

The statistics displayed on the **Buffer Pool Load/Locate Statistics** screen are snapshots of the buffer pool which are refreshed every time you press ENTER.

The following information is displayed on the screen:

| Field | Explanation |
|---|---|
| **Total Locate Calls** | The total number of object location calls; that is, the total number of times the Natural buffer pool manager was requested to search the buffer pool for an object.<br><br>If the location is successful, the object has been found in the buffer pool or the BP cache and need not be loaded from a Natural system file thereby saving calls and I/Os. |
| **Total Locate Calls - successful** | The total number of successful Locate calls as an absolute number. |
| **Total Locate Calls - failed** | The total number of Locate calls that failed. |
| **Quick Locate Calls** | The total number of Quick Locate calls.<br><br>A Quick Locate call is the attempt to locate an object in the buffer pool by using the internal fast locate table. For further information, see *Internal Fast Locate Table*.<br><br>In this case, the library (name, database ID and file number) and the position of the object in the buffer pool of the last successful locate call for this object is used again to locate the object.<br><br>This is the most efficient way to locate an object. |

| Field | Explanation |
|---|---|
| **Quick Locate Calls - successful** | The number of Quick Locate calls that have been successfully performed. |
| **Quick Locate Calls - failed** | The number of Quick Locate calls that failed.<br><br>A failed Quick Locate call indicates that an object could not be located at its previous position in the buffer pool by using the **internal fast locate table** (see the relevant section in *Performance Considerations*).<br><br>This occurs if the object has either been deleted from the buffer pool or was moved to another position. A Quick Locate call that failed results in a Normal Locate call but without a steplib search since the Natural runtime remembers the library that contains the object. |
| **Normal after Quick** | The number of Normal Locate calls that result from **Quick Locate Calls - failed**.<br><br>The value of **Normal after Quick** is always identical to the value of **Quick Locate Calls - failed**. |
| **Normal after Quick - successful** | The number of successful Normal Locate calls that result from **Quick Locate Calls - failed**.<br><br>A Normal after Quick call is successful if the requested object is still in the buffer pool but at another position. |
| **Normal after Quick - failed** | The number of failed Normal Locate calls that result from **Quick Locate Calls - failed**.<br><br>A failed Normal after Quick call indicates that the requested object is no longer available in the buffer pool. As a result, the object is reloaded from the system file by using the library entry in the **internal fast locate table** (see the relevant section in *Performance Considerations*). |
| **Normal Locate Calls** | The total number of Normal Locate calls.<br><br>A Normal Locate call is the attempt to locate an object in the buffer pool without using the **internal fast locate table** (see the relevant section in *Performance Considerations*).<br><br>A Normal Locate call always occurs if an object is referenced for the first time within a Natural session after a `LOGON` system command has been performed. The Natural runtime does not yet know the library where the object resides, compared with a Quick Locate call. |
| **Normal Locate Calls - successful** | The number of Normal Locate calls that were successful in locating the required object in the buffer pool. |
| **Normal Locate Calls - failed** | The number of Normal Locate calls that failed.<br><br>A failed Normal Locate call indicates that an object (identified by its name and the library where it resides) could not be located in the buffer pool. |

| Field | Explanation |
|---|---|
| | A failed Normal Locate will either result in a load from the BP cache or a system file, or a Normal Locate call for the next library in the steplib chain. |
| **STEPLIB Searches** | The number of **Normal Locate Calls** that occurred from failed attempts to find an object in a steplib library.<br><br>Normal Locate calls that perform unsuccessful steplib searches do not result in the load of an object from the BP cache or the system file.<br><br>**STEPLIB Searches** does *not* count Locate calls for objects that are neither contained in the current library nor any steplib or system file (error message `Invalid command, or program does not exist in library`). Locate calls that fail due to incorrect programming add to the counter of **Normal Locate Calls - failed**.<br><br>The number of **STEPLIB Searches** is calculated by using the following formula:<br><br>**Normal Locate Calls - failed** - (**Number Loads into BP** - **Normal after Quick - failed**)<br><br>The fewer the number of **STEPLIB Searches**, the better the buffer pool is performing.<br><br>See also ***Searching in Steplibs***. |
| **Number Loads into BP** | The number of times a load into the buffer pool was performed successfully.<br><br>The load into the buffer pool (storage allocation request) can be triggered either by a load from the database or by a load from the BP cache. |
| **Loads from Cache** | The total number of successful Locate calls of objects that resided in the BP cache. This information is counted only if the previous Locate call (**Normal after Quick - failed** or **Normal Locate Calls - failed**) failed. It indicates the number of database loads saved. This means, that, without the BP cache, the object would have to be loaded from the database. |
| **Loads from DB** | The number of times an object was loaded from a Natural system file into the buffer pool.<br><br>As several load calls may be necessary to load a single object, this value provides the actual number of object loads made since the most recent buffer pool refresh.<br><br>When loading an object, the buffer pool manager uses different search algorithms: see *METHOD=S* and *METHOD=N* in the *Operations* documentation. |

| Field | Explanation |
|---|---|
| **Loads from DB - finished** | The number of object loads that finished successfully.<br><br>An object load cannot finish if the load operation is canceled due to any of the following reasons:<br><br>■ A concurrent object load occurred: see **Loads from DB - concurrent**.<br>■ During the object load, an Adabas response code occurs.<br>■ During the object load, a SYSBPM delete operation is executed for this object. |
| **Loads from DB - concurrent** | The number of object loads that have been performed simultaneously for the same object:<br><br>Concurrent object loads occur if two or more Natural sessions that run simultaneously request the same object. While an object is being loaded by one session, other sessions request the same object and start loading it before a session has finished loading. In this case, the same object is loaded more than once.<br><br>The first session that finishes loading the object will mark the object of the other sessions to be deleted from the buffer pool. The other sessions will then stop loading the object, remove the object marked for deletion from the buffer pool and use the object loaded successfully by the first session.<br><br>The numbers of objects calculated by the counters **Loads from DB - finished** and **Loads from DB - concurrent** are usually identical. The numbers only differ if the concurrent load is only detected after both sessions have finished the load. |
| **Load Calls** | The total number of load calls made since the buffer pool has been refreshed. The load calls are correlated with the access to the system file from which the objects are read.<br><br>The number of system file accesses is calculated as follows:<br><br>■ Adabas system file: The number of **Load Calls** plus the number of **Loads from DB** (see above). The total number does not include Adabas RC calls.<br>■ VSAM system file: The number of **Load Calls**. |
| **Number Loads BP 2nd** | This field is displayed if `METHOD=S` (selection process) is used as the search method for allocating storage.<br><br>This field shows the number of times a storage allocation request satisfied the search criteria of Algorithm 2 as described in *METHOD=S* in the *Operations* documentation. |

| Field | Explanation |
|---|---|
| **Number Load Cycles** | This field is displayed if `METHOD=N` (next available) is used as the search method for allocating storage as described in the *Operations* documentation.<br><br>This field indicates the number of times a search has been performed starting from the top of the buffer pool. This number gives an estimate of the frequency of cycling through the buffer pool in a wrap-around fashion. |
| **Last Cycle Start** | This field is displayed if `METHOD=N` (next available) is used as the search method for allocating storage as described in the *Operations* documentation.<br><br>The time and date when **Number Load Cycles** was last increased. |
| **Number Lock Retries** | This field is displayed if `METHOD=N` (next available) is used as the search method for allocating storage as described in the *Operations* documentation.<br><br>This field indicates the number of times a chain of locked buffer pool entries had to be unlocked, because they could not satisfy the allocation request. |
| **Largest Alloc (TR)** | The largest single allocation size so far requested, specified in number of text records. |
| **Number Load Failure** | The total number of times an object load failed. The reason for a failure is that either all directory entries are in use at the time of the load request or not enough storage is available in the text record section to perform the load. |
| **Number Load Failure - Sizes failing last** | The number of text records that would have been required by the three most recent storage allocation requests that failed. |

For detailed information on the search methods used for allocating space in the buffer pool, see *Buffer Pool Search Methods* in the *Operations* documentation.

## Buffer Pool Fragmentation

This function provides an overview of the buffer pool fragmentation; that is, an overview of how many different Natural objects occupy how many text records, and how the object locations are spread over the buffer pool.

### To invoke Buffer Pool Fragmentation

- In the **Buffer Pool Statistics** menu, enter the following function code:

```
F
```

Or:

Enter the following SYSBPM direct command:

```
DISPLAY FRAGMENTATION
```

The **Buffer Pool Fragmentation** screen appears.

Some of the fields provided on the **Buffer Pool Fragmentation** screen are identical to the items explained in *General Buffer Pool Statistics*:

**Buffer Pool Size**

**Buffer Pool Address**

**Text Record Section**

**Text Record Size**

**Number of Text Records** (corresponds to **Text Records - Total**)

In addition, the screen displays a diagram which shows how many different individual objects occupy how much text record size.

For example:

```
1---+----10---+----20---+----30---+----40---+----50

005F0480
..._.+***______++
....**_.+**_.++_ *..+**+__++++XX
```

Each symbol in the diagram represents one text record, and each sequence of equal symbols represents a different individual object occupying one or more text records. The symbols have the following meaning:

| | |
|---|---|
| _ and . | Objects with a **Current Use Count** (see *Directory Information*) of 0 (zero). |
| + and * | Objects with a **Current Use Count** greater than 0 (zero). |
| blank character | An unused text record. |
| XX | The end of the buffer pool, which means that no further text records are available. |

In the example above, the buffer pool contains 48 text records. Three of them are not in use; the rest is occupied by 24 different objects, 12 of them with a **Current Use Count** of 0 (zero), and 12 with a **Current Use Count** greater than 0.

## Internal Function Usage

This function provides statistical information on the calls made to the Natural buffer pool manager.

### To invoke Internal Function Usage

- In the **Buffer Pool Statistics** menu, enter the following function code:

  ```
  I
  ```

  Or:

  Enter the following SYSBPM direct command:

  ```
  DISPLAY FUNCTION
  ```

  The **Internal Function Usage** screen appears.

The statistics displayed on the **Internal Function Usage** screen are snapshots of the buffer pool which are refreshed every time you press ENTER.

The field **Total Calls** shows the overall number of all internal calls that have been made to the buffer pool manager.

Internally, the buffer pool manager can be invoked for various different functions. For each function, the number of times it has been invoked is displayed, both as an absolute number and as percentage. In addition, these numbers are represented in a horizontal bar chart.

## Buffer Pool Hash Table Statistics

This function only applies to buffer pools of the type Natural.

**Buffer Pool Hash Table Statistics** displays statistics about hash table slots and collisions per slot. The statistics determine the efficiency of the hash algorithm used.

For further information on hash tables, refer to *Buffer Pool Hash Table* in the *Operations* documentation.

The statistics are primarily intended for internal use by Software AG personnel only.

### To invoke Buffer Pool Hash Table Statistics

- In the **Buffer Pool Statistics** menu, enter the following function code:

```
H
```

Or:

Enter the following SYSBPM direct command:

```
DISPLAY HASH
```

The **Hash Table Collisions** screen appears.

The statistics displayed on the **Hash Table Collisions** screen are snapshots of the hash table which are taken every time you press ENTER. The following information is displayed:

| Field | Explanation |
|---|---|
| **Total Number of Slots** | The total number of hash table slots; that is, the total possible entries in the hash table that link the object names with the location of the objects.<br><br>The number of slots, that is, the size of the hash table will be calculated internally depending on the number of text records. |
| **Number of Slots used** | The number of slots in the hash table that have at least one object name mapped to them. |
| **Number of Slots free** | The number of slots in the hash table that have no object name mapped to them. |
| **Max. Collisions per Slot** | The maximum number of collisions of any slot. The maximum number of collisions is the longest possible search path for an object.<br><br>A collision is caused if the name of two different objects is mapped to the same slot by the hash algorithm. In this case, a collision resolution is used in order to find another slot. |
| **Collisions** | The number of current collisions. Depending on the collisions that occur, the table contains up to 10 rows:<br><br>`0` — No collision.<br>`1` — 1 collision.<br>`2` — 2 collisions.<br>`3` — 3 collisions.<br>`4` — 4 collisions.<br>`5` — 5 collisions.<br>`6 - 10` — Between 6 and 10 collisions.<br>`11 - 15` — Between 11 and 15 collisions.<br>`16 - 20` — Between 16 and 20 collisions.<br>`21` — More than 21 collisions. |

| Field | Explanation |
|---|---|
| | No collision means that only one object name is mapped per slot. To locate this object, you need to access the hash table once only.<br><br>If the number of collisions is greater than 0 (zero), for example, *x*, *x*+1 object names are mapped to the same slot. To locate one of these objects, you need to access the hash table up to *x*+1. |
| **Number of Slots** | The number of slots related to the number of collisions.<br><br>In addition, the percentage of these slots related to all slots used is displayed. |
| **Number of Slots Totaled** | The same values as **Number of Slots**, but the values are totaled. |

### Example of Hash Table Statistics

```
 14:36:26
***** NATURAL SYSBPM UTILITY *****
2003-08-13
 BPNAME NATGBP
- Buffer Pool Hash Table Statistics -
Type Global Nat
 BPPROP OFF
Loc DAEF QA41


 Total Number of Slots ..
523


Number of Slots used ..
475 (
90.8 %)
Max. Collisions


Number of Slots free ..
48 (
9.1 %)
per Slot ..... 7




Collisions
Number of Slots
Number of Slots Totaled


0
0 (
0.0 %)
0 (
0.0 %)


1
164 (
34.5 %)
164 (
34.5 %)


2
194 (
40.8 %)
358 (
```

```
75.3 %)


3
96 (
20.2 %)
454 (
95.5 %)


4
16 (
3.3 %)
470 (
98.9 %)


5
4 (
0.8 %)
474 (
99.7 %)


6 - 10
1 (
0.2 %)
475 ( 100.0 %)









Command ===>

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

Help
Exit
Last
Flip
Canc
```

# Performance Hints

This function only applies to a buffer pool and a BP cache of the type Nat (Natural).

The **Performance Hints** function provides statistical information about workloads on a Natural buffer pool and a BP cache including ratings of the overall performance.

**Related Topic:**

- *Performance Considerations*

This section covers the following topics:


- Invoking Performance Hints
- Evaluating Performance Hints


## Invoking Performance Hints

This section describes how to invoke the **Performance Hints** function and the statistics field displayed on the **Performance Hints** screen.

### To invoke Performance Hints

- In the **Buffer Pool Statistics** menu, enter the following function code:

  ```
  P
  ```

  Or:

  Enter the following SYSBPM direct command:

  ```
  DISPLAY PERFORMANCE
  ```

  A **Performance Hints** screen similar to the example below appears:

```
13:27:16
***** NATURAL SYSBPM UTILITY *****
2005-05-19

BPNAME QA41GBP
- Performance Hints -
Type Global Nat

BPPROP OFF
Loc DAEF QA41

Preload QA41GBPL



Rating


(1=best - 6=worst)


Buffer Pool


Locates / Loads Ratio ......
162.85
3


Wrap Time Last (hh:mm:ss) ..
00:06:29
4


Wrap Time Avg (hh:mm:ss) ...
00:01:22
5




BP Cache


Object Reuse Factor ........
3.11
4


Wrap Time Last (hh:mm:ss) ..
```

```
02:02:29
1


Wrap Time Avg (hh:mm:ss) ...
00:32:06
3


Get / Search % .............
74.48 %









 Command ===>

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

Help
Exit
Last
Flip
Canc
```

The fields on the **Performance Hints** screen provide the following information:

| Field | Explanation |
|---|---|
| **Buffer Pool - Locates / Loads Ratio** | The ratio of **Total Locate Calls - successful** to **Loads from DB**. A value greater than 1 indicates that Natural located more objects in the buffer pool than it loaded from the system file.<br><br>This ratio serves as a buffer pool efficiency indicator. The larger the number, the better the buffer pool is performing. This is the primary indicator of performance from one buffer pool session to the next. |
| **Buffer Pool - Wrap Time Last** | This field is displayed if `METHOD=N` (next available) is used as the search method for allocating storage as described in the *Operations* documentation.<br><br>The time in hours, minutes and seconds (*hh*:*mm*:*ss*) since the buffer pool was last completely reused (wrapped around).<br><br>Objects are loaded into the buffer pool one after the other (in sequential order) filling the top of the buffer pool first and the bottom last. When the bottom of the buffer pool has |

| Field | Explanation |
|---|---|
| | been reached, the buffer end is wrapped around and the next object is loaded again at the top of the buffer pool.<br><br>When the buffer pool has been filled completely for the first time, a new object loaded into the buffer overwrites an object that was loaded in the previous wrap-around cycle and that is not currently locked (marked as resident or in use).<br><br>Every object loaded into the buffer pool is assigned a directory entry that contains information such as the name of the object, the library where it is stored and the **BP Load Time** time stamp of the loading into the buffer pool.<br><br>**Wrap Time Last** is evaluated each time an object is loaded into the buffer pool. **Wrap Time Last** indicates the time period between the last loading of an object (**BP Load Time**) and the last overwriting of an object.<br><br>The longer the period of a wrap-around cycle, the better the buffer pool is performing. This period can vary considerably at night or over a weekend when user traffic is low compared with normal working hours. |
| **Buffer Pool - Wrap Time Avg** | This field is displayed if `METHOD=N` (next available) is used as the search method for allocating storage as described in the *Operations* documentation.<br><br>The average time in hours, minutes and seconds (*hh*:*mm*:*ss*) of one wrap-around cycle since the buffer pool was initialized or refreshed.<br><br>**Wrap Time Avg** is calculated by dividing the life time of the buffer pool by the number of wrap-around cycles.<br><br>**Wrap Time Last** compared with **Wrap Time Avg** shows whether the buffer pool is currently being used more frequently than on average. |
| **BP Cache - Object Reuse Factor** | The ratio of objects swapped out of the BP cache (Get calls) to objects swapped into the BP cache (**Put calls**).<br><br>The value is calculated by dividing successful Get calls by successful Put calls. It shows the overall reuse factor; that is, how often an object loaded once into the BP cache could be successfully reloaded into the buffer pool. The higher the value, the better the BP cache efficiency.<br><br>Example of an **Object Reuse Factor**:<br>A ratio of `5.70` indicates that an object loaded in the BP cache was swapped into the buffer pool 5.7 times on average. |
| **BP Cache - Wrap Time Last** | The time in hours, minutes and seconds (*hh*:*mm*:*ss*) since the BP cache was last completely reused (wrapped around).<br><br>Objects are loaded into the BP cache one after the other (in sequential order) filling the top of the BP cache first and the bottom last. When the bottom of the BP cache has been reached, the end of the BP cache is wrapped around and the next object is loaded again at the top of the BP cache. When the BP cache has been filled completely for the first time, a new object loaded into the buffer overwrites the object that was loaded in the previous wrap-around cycle. |

| Field | Explanation |
|---|---|
| | Each object loaded into the BP cache is assigned a directory entry that contains information such as the name of the object, the library where it is stored and the **BPC Load Time** time stamp when it was loaded into the BP cache.<br><br>**Wrap Time Last** is evaluated every time an object is loaded into the BP cache. **Wrap Time Last** indicates the time period between the last loading of an object (**BPC Load Time**) and the last overwriting of an object.<br><br>The longer the period of a wrap-around cycle, the better the BP cache is performing. This period can vary considerably at night or over a weekend when user traffic is low compared with normal working hours. |
| **BP Cache - Wrap Time Avg** | The average time in hours, minutes and seconds (*hh*:*mm*:*ss*) of one wrap-around cycle since the BP cache was started.<br><br>**Wrap Time Avg** is calculated by dividing the life time of the BP cache by the number of wrap-around cycles.<br><br>**Wrap Time Last** compared with **Wrap Time Avg** shows whether the BP cache is currently being used more frequently than on average. |
| **BP Cache - Get / Search %** | The percentage of objects returned from the BP cache successfully (Get calls) compared with the total number of search requests (Search calls) the buffer pool sent to the BP cache.<br><br>This value indicates the percentage of objects the buffer pool could load from the BP cache, instead of the Natural system file FNAT or FUSER. The higher the value, the better the cache efficiency.<br><br>Keep in mind that a search for an object in a chain of steplibs may increase the number of Search calls the buffer pool sends to the BP cache. However, these calls will not result in a successful Get call or a load from the Natural system file, because an object may not be found in a steplib. **Get / Search %** does not consider the search through a long chain of steplibs that is frequently used. We recommend that you use as few steplibs as possible to increase overall performance.<br><br>Example of a **Get / Search %** value:<br>A value of 70% indicates that 70 % of all objects loaded into the buffer pool were retrieved from the BP cache and 30% were loaded from the system file. |

## Evaluating Performance Hints

The statistical values provided on the **Performance Hints** screen are the basis for a performance rating system where 1 denotes best (highest) and 6 denotes worst (lowest) performance.

The ratings shall give you an idea of how a buffer pool or a BP cache performs with the sized specified for them. The rating values should suit the requirements of most system environments.

There are environments where it has proven to be worth having a BP cache that is five times as big as the buffer pool. If ratings tend to be bad, the size of the buffer pool or BP cache should be increased. However, the size of a buffer pool or a BP cache can guarantee good performance even

though ratings are bad. If the ratings of a BP cache are good, bad ratings for the buffer pool do not carry so much weight. For environments with extreme workloads, the ratings can, however, be useful indicators for when the size of the buffer pool or the BP cache should be changed.

The statistical values displayed on the **Performance Hints** screen are snapshots of the Natural buffer pool and BP cache which are refreshed each time a wrap-around occurs. Buffer pool and the BP cache should run for some time and the statistics values reach a certain extent to obtain meaningful results from these values. For example, if the size of a buffer pool is so large that only very few BP cache calls are required, the statistics regarding the BP cache will not be meaningful.

When evaluating the statistics you should also consider the type of system environment (for example, production versus test), the type of applications using the buffer pool (batch, online, user-defined or system), user traffic (peak hours or normal hours) and extraordinary operational factors.

## PF Keys and Direct Commands

On the buffer pool statistics screens, you can use the PF keys or SYSBPM direct commands listed in the table below. An underlined portion of a command represents its minimum abbreviation. For further commands, see ***SYSBPM Direct Commands***.

| PF Key | Command | Function |
|---|---|---|
| PF1 | | Provides SYSBPM help information: see also ***Online Help***. |
| PF3 | EXIT | Leaves the current function/screen and displays the previous screen. |
| PF4 | LAST | Displays the SYSBPM direct command entered most recently. |
| PF6 | FLIP | Switches the PF-key line: toggles between the display of PF1 to PF12 and PF13 to PF24. |
| PF8 (Load) | DISPLAY LOAD | Only applies to the screen **General Buffer Pool Statistics**.<br><br>Displays the **Buffer Pool Load/Locate Statistics** screen. |
| PF8 (Gen) | DISPLAY GENERAL | Only applies to the **Buffer Pool Load/Locate Statistics** screen.<br><br>Displays the **General Buffer Pool Statistics** screen. |
| PF12 | CANCEL | Same as EXIT. |
| PF15 | MENU | Invokes the SYSBPM **Main Menu**. |

# 60 BP Cache Statistics


- General BP Cache Statistics ..... 380
- BP Cache Call Statistics ..... 382
- BP Cache Hash Table Statistics ..... 384
- Performance Hints ..... 385
- PF Keys and Direct Commands ..... 385

The **BP Cache** function only applies to a buffer pool of the type Natural.

This function invokes the **BP Cache Statistics** menu which is used to obtain statistical information on the BP cache.

Note that the **BP Cache** function can only be executed if a BP cache has been installed when initializing a global buffer pool (no BP cache support for local buffer pools).

### To invoke BP Cache Statistics

- In the SYSBPM **Main Menu**, enter the following function code:

  ```
  C
  ```

  Or:

  Enter the following SYSBPM direct command:

  ```
  DISPLAY CSTATISTICS
  ```

  The **BP Cache Statistics** menu appears.

The functions available in the **BP Cache Statistics** menu and the commands provided on the screens that are invoked by these functions are explained in this section.

## General BP Cache Statistics

This function displays addresses and statistics regarding the activity of the BP cache.

### To invoke General BP Cache Statistics

- In the **BP Cache Statistics** menu, enter the following function code:

  ```
  G
  ```

  Or:

  Enter the following SYSBPM direct command:

  ```
  DISPLAY CGENERAL
  ```

  The **General BP Cache Statistics** screen appears.

The statistics displayed on the **General BP Cache Statistics** screen are snapshots of the buffer pool, which are refreshed each time you press ENTER. The following information is displayed:

| Field | Explanation |
|---|---|
| **Dataspace - Name** | The name of the dataspace where the BP cache resides. |
| **Dataspace - SToken** | The term SToken (for Space Token) identifies a dataspace. |
| **Dataspace - ALET** | The term ALET (for Address List Entry Token) identifies an index for accessing the dataspace. |
| **Dataspace - Size (MB)** | The size of the BP cache in MB. |
| **Dataspace - Current state** | The status of the BP cache:<br><br>`not initialized`<br>`locked for init`<br>`closed`<br>`free for operation`<br>`undefined` |
| **Dataspace - Initialization** | The date and time when the BP cache was initialized. |
| **Internal buffer offsets - Header buffer** | The header of the BP cache which contains general BP cache information. |
| **Internal buffer offsets - Hash buffer** | Contains the hash table (see also ***BP Cache Hash Table Statistics***). |
| **Internal buffer offsets - Directory buffer** | The address of the BP cache directory section relative to the beginning of the BP cache.<br><br>Each Natural object loaded in the BP cache requires a directory entry that contains information on this object. The space for these directory entries is acquired from the BP cache itself. |
| **Internal buffer offsets - Text buffer** | The address of the text buffer relative to the beginning of the BP cache.<br><br>After allocating the space for all other buffers, the remaining space is divided into text records with a size of 4 KB.<br><br>An object can occupy one or more text records, depending on its size. |
| **Tot. Text Records** | The total number of text records in the BP cache.<br><br>The number of text records depends on the BP cache size. The text record size for the BP cache is 4 KB. |
| **Insert position** | The index number of the text record into which the next object will be inserted.<br><br>Objects will be inserted into the BP cache when they have to be removed from the buffer pool. |
| **Reuse cycles** | The number of times the BP cache has been completely reused.<br><br>Every time the BP cache is full, the BP cache manager reuses the BP cache from the start and overwrites the object(s) from there. The objects will remain in the BP cache until the BP cache is used again. |

| Field | Explanation |
|---|---|
| **Objects - Max Loaded** | The maximum number of objects currently loaded in the BP cache. |
| **Objects - Loaded** | The number of objects currently loaded in the BP cache. |

## BP Cache Call Statistics

This function provides statistical information on the loading (put), retrieving (get) and deleting of objects into/from the BP cache. This information also serves as an indicator of BP cache performance.

### To invoke BP Cache Call Statistics

- In the **BP Cache Statistics** menu, enter the following function code:

  ```
  L
  ```

  Or:

  Enter the following SYSBPM direct command:

  ```
  DISPLAY CLOAD
  ```

  The **BP Cache Call Statistics** screen appears.

The statistics displayed on the **BP Cache Call Statistics** screen are snapshots of the buffer pool which are refreshed each time you press ENTER. The following information is displayed:

| Field | Explanation |
|---|---|
| **Search calls** | The number of Search calls the buffer pool sent to the BP cache while attempting to find an object in the BP cache.<br><br>If an object is found, a Search call results in a Get call. |
| **Get calls (from BP cache)** | The number of Get calls the buffer pool sent to the BP cache while attempting to load an object from the BP cache into the buffer pool. |
| **Get calls - successful** | The number of successful Get calls the BP cache performed, that is, the number of objects the BP cache swapped into the buffer pool.<br><br>A Get call is successful if an object the buffer pool attempted to load is actually loaded from the BP cache into the buffer pool.<br><br>A Get call is unsuccessful, for example, if an object was deleted after it was found by the Search call. |
| **Put calls (to BP cache)** | The number of Put calls the buffer pool sent to the BP cache while attempting to swap an object from the buffer pool into the BP cache. |

| Field | Explanation |
|---|---|
| **Put calls - successful** | The number of Put calls that resulted in an object to swapped from the buffer pool into the BP cache. |
| **Put calls - obj. already cached** | The number of Put calls the buffer pool sent to the BP cache for objects that were already loaded in the BP cache. |
| **Delete calls** | The number of Delete calls the buffer pool sent to the BP cache while attempting to delete an object from the BP cache.<br><br>A Delete call requests either a single object or a range of objects (see also the section ***Delete Objects***). |
| **Delete calls - successful** | The number of successful Delete calls the buffer pool sent to the BP cache.<br><br>A Delete call is successful if at least one object is actually deleted from the BP cache.<br><br>A Delete call is unsuccessful if the object requested was not loaded in the BP cache and hence could not be deleted.<br><br>Compared with the total number of Delete calls, the number of successful Delete calls can be very low. This happens, for example, if several Natural objects are cataloged with the `CATALL` command. In this case, for every object cataloged successfully, Natural sends a Delete call to the BP cache. However, at the time the Delete call is sent, most of the cataloged objects are usually not loaded in the BP cache and the delete attempt fails. |
| **Nbr objects deleted - by roll-over** | The number of Natural objects which have been deleted due to a full cache before new objects could be loaded. |
| **Nbr objects deleted - by command** | The number of Natural objects, which have been deleted by the SYSBPM function all or by Natural commands or utilities such as CAT, STOW, CATALL or SYSMAIN.<br><br>**Note:** You can delete multiple objects with `Delete call`. |
| **Initialization** | The date and time when the BP cache was initialized. |
| **Last reuse cycle** | The load date and time of the object that was most recently overwritten.<br><br>An object is overwritten in the BP cache when its space has to be reused in order to load another object. The object that was loaded first in the BP cache will be swapped first. This means the load date and time of the object that has been in the BP cache longest corresponds to the date and time in **Last reuse cycle**. |
| **Last access** | The date and time when the buffer pool last accessed the BP cache. |
| **Last Put (to BP cache)** | The date and time when the buffer pool last sent a Put call to the BP cache. |
| **Last Get (from BP cache)** | The date and time when the buffer pool last sent a Get call to the BP cache. |
| **Last Delete** | The date and time when the buffer pool last sent a Delete call to the BP cache. |

# BP Cache Hash Table Statistics

This function displays statistics about hash table slots and collisions per slot. The statistics determine the efficiency of the hash algorithm used.

### To invoke BP Cache Hash Table Statistics

- In the **BP Cache Statistics** menu, enter the following function code:

  ```
  H
  ```

  Or:

  Enter the following SYSBPM direct command:

  ```
  DISPLAY CHASH
  ```

  The **Cache Hash Table Collisions** screen appears.

The statistics displayed on the **Cache Hash Table Collisions** screen are snapshots of the hash table which are refreshed every time you press ENTER. The following information is displayed:

| Field | Explanation |
|---|---|
| **Total Number of Slots** | The total number of hash table slots; that is, the total possible entries that link the object name with the location of the object.<br><br>The number of slots, that is, the size of the hash table will be calculated internally depending on the number of text records. |
| **Number of Slots used** | The number of slots that have one or more entries. |
| **Number of Slots free** | The number of slots that have no entry. |
| **Max. Collisions per Slot** | The maximum number of collisions of all slots.<br><br>The maximum number of collisions is the longest possible search path for an object. |
| **Collisions** | The number of possible collisions.<br><br>`0` (zero) means no collision or one entry. When there are more than 5 collisions, the number of collisions will be specified in ranges (for example, `6 - 10`). |
| **Number of Slots** | The number of slots grouped by their number of collisions.<br><br>For example, if the number of collisions is `3`, the search algorithm must side step a maximum of 3 times to find an object. In addition, the percentage of these slots related to all slots used is displayed. |
| **Number of Slots Totaled** | The same values as **Number of Slots**, but the values are totaled. |

## Performance Hints

See ***Performance Hints*** in the section *Buffer Pool Statistics*.

## PF Keys and Direct Commands

On BP cache statistics screens, you can use the PF keys or SYSBPM direct commands listed in the table below. An underlined portion of a command represents its minimum abbreviation. For further commands, see ***SYSBPM Direct Commands***.

| PF Key | Command | Function |
|---|---|---|
| PF1 | | Provides SYSBPM help information: see also ***Online Help***. |
| PF3 | EXIT | Leaves the current function/screen and displays the previous screen. |
| PF4 | LAST | Displays the SYSBPM direct command entered most recently. |
| PF6 | FLIP | Switches the PF-key line: toggles between the display of PF1 to PF12 and PF13 to PF24. |
| PF8 (CLoad) | DISPLAY CLOAD | Only applies to the **General BP Cache Statistics** screen.<br><br>Displays the **BP Cache Call Statistics** screen. |
| PF8 (CGen) | DISPLAY CGENERAL | Only applies to the **BP Cache Call Statistics** screen.<br><br>Displays the **General BP Cache Statistics** screen. |
| PF12 | CANCEL | Same as EXIT. |
| PF15 | MENU | Invokes the SYSBPM **Main Menu**. |

# 61 Message Pool Statistics

The **SYSBPM** function invokes the **Message Pool Statistics** menu which is used to obtain message-pool-related statistics.

### To invoke Message Pool Statistics

- In the SYSBPM **Main Menu**, enter the following function code:

  ```
  M
  ```

  Or:

  Enter the following SYSBPM direct command:

  ```
  DISPLAY MSTATISTICS
  ```

  The **Message Pool Statistics** screen appears.

The statistics displayed on the **Message Pool Statistics** screen are snapshots of the message pool which are refreshed each time you press ENTER. The following information is displayed:

| Field | Explanation |
|---|---|
| **Message Pool Size (MB)** | The size of the whole message pool in MB. The message pool size can be specified with the `NTBPI` macro in the parameter module or with the `BPI` profile parameter described in the *Parameter Reference* documentation. |
| **Available text entries** | The maximum number of texts the message pool can contain. |
| **Free text entries** | Number of additional texts which may be loaded into the message pool. |
| **Start date** | The date when the message pool was originally started. |
| **Start time** | The time when the message pool was originally started. |
| **Full conditions** | The number of times a message could not be added to the message pool because there were no free text entries left. |
| **Total number of locates** | The total number of message retrieves from the message pool. |

| Field | Explanation |
| --- | --- |
| **Successful direct locates** | The number of times a message could be retrieved without using the hash table. |
| **Collisions high water** | The maximum number of collisions. In other word, the maximum number of texts using the same entry in the hash table. |

# 62 Select Buffer Pool


- Invoking Select Buffer Pool ..... 390
- Displaying Buffer Pools ..... 391
- Resetting a Buffer Pool ..... 391

SYSBPM provides functions for displaying and administrating the buffer pools (including the BP cache) defined for your Natural system environment. A buffer pool not defined on the current Natural session at startup needs to be selected before it can be administrated.

## Invoking Select Buffer Pool

### To invoke Select Buffer Pool

1 In the SYSBPM **Main Menu**, enter **Code** `S`.

Or:

Enter the following SYSBPM direct command:

```
SELECT BP
```

The **Select Buffer Pool** window appears with the following information on your current buffer pool (global or local) and on all other global buffer pools currently available in your Natural system environment:

| | |
|---|---|
| **BPNAME** | The name of the buffer pool. |
| **Type** | The type of the buffer pool such as `Global Nat`, `Local Nat`, `Global Sort`, `Global DL/I`, `Edit` (Editor) or `Mon` (Monitor). |
| **Status** | The current status. |
| **Preload** | The name of the preload list (if loaded). |
| **Address** | The address of the buffer pool. |
| **Loc** | The location of the buffer pool indicated by **HostID** and **SubsID** (subsystem ID). |

For further information on the fields, see *SYSBPM Main Menu - Fields, Functions and Commands*.

2 In column **C**, enter any character in front of the buffer pool that you want to select and press ENTER. Note that only buffer pools of type `Natural`, `DL/I` or `Sort` can be selected, buffer pools of another type can only be displayed. The specified buffer pool is now defined in your current session and the contents of the buffer pool-related fields (as mentioned above) displayed on top of the SYSBPM **Main Menu** is changed accordingly.

Once you have selected a buffer pool from the **Select Buffer Pool** window, all SYSBPM functions apply to this buffer pool. Your Natural session itself, however, will continue to run with the buffer pool that was used when the session was started.

## Displaying Buffer Pools

### To display the buffer pools available in your Natural system

- Enter the following SYSBPM direct command:

  ```
  DISPLAY BUFFERPOOL
  ```

  Or:

  Perform **the first step** of the instructions on *Invoking Select Buffer Pool*

  The **Display Buffer Pools** window appears which provides the same information as described for the **Select Buffer Pool window**.

## Resetting a Buffer Pool

### To reset a buffer pool

- Enter the following SYSBPM direct command:

  ```
  RESET BUFFERPOOL
  ```

  SYSBPM switches back to the buffer pool originally defined for your current Natural session and the contents of the buffer pool-related fields (as mentioned in *Invoking Select Buffer Pool*) displayed on top of the SYSBPM **Main Menu** change accordingly.

  Or:

  Invoke the **Select Buffer Pool function** (described earlier) and select the startup buffer pool.

# 63 Select Message Pool


- Invoking Select Message Pool ..... 394
- Displaying Message Pools ..... 395
- Resetting a Message Pool ..... 395

# Invoking Select Message Pool

### To invoke Select Message Pool

1 In the SYSBPM **Main Menu**, enter **Object Pool** `M` and **Code** `S`.

Or:

Enter the following SYSBPM direct command:

```
SELECT MP
```

The **Select Message Pool** window appears with the following information on your current message pool and on all other message pools currently available in your Natural system environment:

| | |
|---|---|
| **BPNAME** | The name of the message pool. |
| **Type** | `MSG`, the type for message pools. |
| **Preload** | The name of the preload list (if loaded). |
| **ALET** | The term ALET (for Address List Entry Token) identifies an index for accessing the dataspace. |
| **Loc** | The location of the buffer pool indicated by **HostID** and **SubsID** (subsystem ID). |

For further information on the fields, see *SYSBPM Main Menu - Fields, Functions and Commands*.

2 In column **C**, enter any character in front of the message pool that you want to select and press ENTER. The specified message pool is now defined in your current session and the contents of the message pool-related fields (as mentioned above) displayed on top of the SYSBPM maps is changed accordingly. Note that only the functions **L** (**List Objects**), **D** (**Delete Objects**), and **M** (**Message Pool Statistics**) or their respective direct command are available.

Once you have selected a message pool from the **Select Message Pool** window, all SYSBPM functions apply to this message pool. Your Natural session itself, however, will continue to run with the message pool that was used when the session was started.

## Displaying Message Pools

### To display the message pools available in your Natural system environment

- Enter the following SYSBPM direct command:

  ```
  DISPLAY MP
  ```

  Or:

  Perform **the first step** of the instructions on *Invoking Select Message Pool*

  Or:

  The **Display Message Pool** window appears which provides the same information as described for the **Select Message Pool** window.

## Resetting a Message Pool

### To reset a message pool

- Enter the following SYSBPM direct command:

  ```
  RESET MP
  ```

  SYSBPM switches back to the message pool originally defined for your current Natural session and the contents of the message pool-related fields (as mentioned in *Invoking Select Message Pool*) displayed on top of the SYSBPM maps is changed accordingly.

  Or:

  Invoke the **Select Message Pool function** (described earlier) and select the startup message pool.

# 64 Blacklist Maintenance


- Maintain Blacklist ..... 398
- List Object Sets ..... 402
- Edit Object Set ..... 402
- Add Object Set to Blacklist ..... 405
- Delete Object Set from Blacklist ..... 406
- Delete Object Set Source Object ..... 407
- Additional Object Set Maintenance with Utilities ..... 407
- Blacklist Maintenance in Batch Mode ..... 408

This function is used to maintain a blacklist of Natural objects. In a blacklist, you can specify the Natural objects that are not to be loaded into the buffer pool; objects that are already loaded in the buffer pool are then deleted. If the BP cache is enabled, the Natural objects will also be deleted from the BP cache. The blacklist always applies to the buffer pool currently active.

In a blacklist, you can maintain individual Natural objects and/or libraries and object sets which contain multiple Natural objects. In an object set, you specify the objects not to be executed and add a single set (instead of multiple individual objects) to the blacklist. You can also combine both: maintain objects individually or by sets.

The source code of an object set is stored as a Natural source object of the type Text in the current Natural library and system file. The first line of an object set source contains the comment `**BBL**` (buffer pool blacklist).

For further information on the blacklist, see the relevant section in *Natural Buffer Pool* in the *Operations* documentation.

### To invoke Blacklist Maintenance

- In the SYSBPM **Main Menu**, enter the following function code:

  ```
  B
  ```

  Or:

  Enter the following SYSBPM direct command:

  ```
  BLACKLIST
  ```

  The **Blacklist Maintenance** menu appears.

The functions provided in the **Blacklist Maintenance** menu are explained in the following section. This section also provides information on additional blacklist maintenance functions.

## Maintain Blacklist

This function invokes the **Maintain Blacklist** screen where you can display and maintain all Natural objects currently available in the blacklist.

### To invoke the Maintain Blacklist screen

- In the **Blacklist Maintenance** menu, enter the following function code:

  ```
  M
  ```

  Or:

Enter the following SYSBPM direct command:

```
DISPLAY BLACKLIST
```

The **Maintain Blacklist** screen appears with the current blacklist. Press PF7 to scroll one page backward and PF8 to scroll one page forward.

Depending on the mode set when calling the **Maintain Blacklist** function earlier during a SYSBPM session, the **Maintain Blacklist** screen appears in **Display Mode** (default when initializing SYSBPM) or **Add Mode**. Use PF9 to switch from one mode to the other.

This section covers the following topics:

- Adding Objects
- Modifying Objects
- Deleting Objects

## Adding Objects

### To add objects to the blacklist

1 In the **Blacklist Maintenance** menu, enter the following function code:

```
M
```

Or:

Enter the following SYSBPM direct command:

```
DISPLAY BLACKLIST
```

The **Maintain Blacklist** screen appears.

2 If required, press PF9 to switch to **Add Mode**.

A screen with empty input fields appears.

3 In the relevant selection fields, enter the name of the library where the objects are stored, the names of the objects and the corresponding database IDs (DBID) and file numbers (FNR). If DBID and FNR are left blank, they will be taken from the current system file FUSER or FNAT in libraries whose names start with SYS (except the library SYSTEM).

If you need to clear the **Add Mode** screen, in the Command line, enter either of the following:

```
CLE
```

or

```
CLEAR
```

4 Press PF5 to confirm the addition.

Or:

In the Command line, enter either of the following:

```
UP
```

or

```
UPDATE
```

An appropriate message appears.

## Modifying Objects

### To modify objects on the blacklist

1 In the **Blacklist Maintenance** menu, enter the following function code:

```
M
```

Or:

Enter the following SYSBPM direct command:

```
DISPLAY BLACKLIST
```

The **Maintain Blacklist** screen appears.

2 If required, press PF9 to switch to **Display Mode** and obtain the list of all objects currently contained in the blacklist.

3 In the relevant input field(s), replace the existing entries with new values.

4 Press PF5 to confirm the modification.

Or:

In the Command line, enter either of the following:

```
UP
```

or

```
UPDATE
```

An appropriate message appears.

## Deleting Objects

### To delete individual objects from the blacklist

1 In the **Blacklist Maintenance** menu, enter the following function code:

```
M
```

Or:

Enter the following SYSBPM direct command:

```
DISPLAY BLACKLIST
```

The **Maintain Blacklist** screen appears.

2 If required, press PF9 to switch to **Display Mode** and obtain a list of all objects currently on the blacklist.

3 In column **C**, next to the object(s) required, enter the following line command:

```
DE
```

4 Press ENTER to confirm the deletion.

An appropriate message appears.

### To delete all objects from the blacklist

1 In the **Blacklist Maintenance** menu, enter the following function code:

```
M
```

Or:

Enter the following SYSBPM direct command:

```
DISPLAY BLACKLIST
```

The **Maintain Blacklist** screen appears.

2 Press PF2.

The **Confirm Delete** window appears.

3 Confirm the deletion, by entering `Y` (Yes).

Or:

Cancel the deletion, by entering `N` (No; this is the default) or by pressing PF3 without entering anything in the window.

4 Press ENTER to confirm the action.

An appropriate message appears.

## List Object Sets

This function invokes the **List Object Sets** screen which displays a list of all existing object sets.

### To invoke the List Object Sets screen

- In the **Blacklist Maintenance** menu, enter function code `L`, a library name and an object set name. Asterisk (*) notation is also allowed for an object set name.

  Or:

  Enter the following SYSBPM direct command:

  ```
  LIST SET library-name set-name
  ```

  Asterisk (*) notation is also allowed for *set-name*.

  The **List Object Sets** screen appears with the specified set(s).

You can manipulate an object set from the **List Object Sets** screen by using any of the line commands provided to modify a set and add it to or delete it from the blacklist. For a list of possible line commands, enter a question mark (?) in the leftmost screen column (prefix information).

For a list of commands that can be entered in the command line of the **List Object Sets** screen, invoke the **Help** window by entering a question mark (?) in the command line. If required, press PF7 to scroll backward and PF8 to scroll forward in the window.

## Edit Object Set

This function invokes the **Edit Object Set** screen where you can create a new object set, add objects to an existing set or modify them, or delete objects from a set.

The editing functions provided on the **Edit Object Set** screen are a subset of the functions provided by the Software AG Editor described in the *Editors* documentation.

For a list of available line commands, enter a question mark (?) in the leftmost screen column (prefix information).

For a list of commands that can be entered in the command line of the **Edit Object Set** screen, invoke the **Help** window by entering a question mark (?) in the command line. If required, press PF7 to scroll backward and PF8 to scroll forward in the window.

This section covers the following topics:

- Creating Object Sets
- Modifying Object Sets

## Creating Object Sets

### To create an object set

1 In the **Blacklist Maintenance** menu, enter function code E and a library name. Do *not* enter the name of an object set but clear the contents (if any) of the corresponding field.

The **Edit Object Set** screen appears.

2 In the relevant input fields, enter the name of the library where the objects are stored, the names of the objects and the corresponding database IDs (DBID) and file numbers (FNR). If DBID and FNR are left blank, they will be taken from the current system file FUSER or FNAT in libraries whose names start with SYS (except the library SYSTEM).

3 To save the object set as a source object of the type Text, in the command line of the **Edit Object Set** screen, enter the following:

```
SA set-name
```

## Modifying Object Sets

This section provides instructions for adding objects to an object set, modifying existing objects and deleting objects from an object set. Note that any of these object set modifications will *not* update the current blacklist.

### To add a new object to an object set

1 In the **Blacklist Maintenance** menu, enter function code E, a library name and an object set name.

Or:

On the **List Object Sets** screen, in the leftmost column, next to the object set required, enter the following line command:

```
E
```

Or:

Enter the following SYSBPM direct command:

```
EDIT SET library-name set-name
```

The **Edit Object Set** screen appears with the specified object set.

2 Complete the input fields by entering the name of the library where the objects are stored, the names of the objects and the corresponding database IDs (DBID) and file numbers (FNR). If DBID and FNR are left blank, they will be taken from the current system file FUSER or FNAT in libraries whose names start with SYS (except the library SYSTEM).

3 To save the modifications, in the command line of the **Edit Object Set** screen, enter the following:

```
SA
```

### To modify an object of an object set

1 In the **Blacklist Maintenance** menu, enter function code `E`, a library name and an object set name.

Or:

On the **List Object Sets** screen, in the leftmost column, next to the object set(s) required, enter the following line command:

```
E
```

Or:

Enter the following SYSBPM direct command:

```
EDIT SET library-name set-name
```

The **Edit Object Set** screen appears with the specified object set.

2 In the relevant input field(s), replace the existing entries with new values.

3 To save the modifications, in the command line of the **Edit Object Set** screen, enter the following:

```
SA
```

### To delete an object from an object set

1 In the **Blacklist Maintenance** menu, enter function code `E`, a library name and an object set name.

Or:

On the **List Object Sets** screen, in the leftmost column, next to the object set(s) required, enter the following line command:

```
E
```

Or:

Enter the following SYSBPM direct command:

```
EDIT SET library-name set-name
```

The **Edit Object Set** screen appears with the specified object set.

2 In the leftmost column, next to the object required, enter the following line command:

```
D
```

The specified object is removed from the object set.

3 To save the modification, in the command line of the **Edit Object Set** screen, enter the following:

```
SA
```

## Add Object Set to Blacklist

This function is used to add all objects of an object set to the blacklist.

### To add an object set to the blacklist

- In the **Blacklist Maintenance** menu, enter function code `A`, a library name and an object set name.

  Or:

  On the **List Object Sets** screen, in the leftmost column, next to the object set(s) required, enter the following line command:

  ```
  AC
  ```

  Or:

  On the **Edit Object Set** screen, in the command line, enter the following:

  ```
  AC
  ```

  Or:

  Enter the following SYSBPM direct command:

```
ADD SET library-name set-name
```

A message appears confirming that the object set was added to the blacklist.

**Note:** The command `AC` denotes `ACTIVATE` which is the equivalent of **Add Object Set to Blacklist**.

## Delete Object Set from Blacklist

This function is used to delete all objects of an object set from the blacklist. Note that the **Delete Object Set** function will *not* delete the object set as a source object. The objects of the object set can be added to the blacklist again at any time, as described above. See also *Delete Object Set Source Object*.

### To delete an object set from the blacklist

- In the **Blacklist Maintenance** menu, enter function code `D`, a library name and an object set name.

  Or:

  On the **List Object Sets** screen, in the leftmost column, next to the object set(s) required, enter the following line command:

  ```
  DA
  ```

  Or:

  On the **Edit Object Set** screen, in the command line, enter the following:

  ```
  DA
  ```

  Or:

  Enter the following SYSBPM direct command:

  ```
  DELETE SET library-name set-name
  ```

  A message appears confirming that the object set was deleted from the blacklist.

**Note:** The command `DA` denotes `DEACTIVATE` which is the equivalent of **Delete Object Set from Blacklist**.

## Delete Object Set Source Object

**To delete the source object of an object set**

1 In the **Blacklist Maintenance** menu, enter function code `L`, a library name and an object set name.

   Or:

   Enter the following SYSBPM direct command:

   ```
   LIST SET library-name list-name
   ```

   The **List Object Sets** screen appears.

2 In the leftmost column, next to the object set required, enter the following line command:

   ```
   D
   ```

   The **DELETE** window appears.

3 Confirm the deletion by entering the name of the object set.

   A confirmation message appears.

**Note:** Deleting the source object of an object set will *not* update the current blacklist.

## Additional Object Set Maintenance with Utilities

The Natural utilities SYSMAIN and Object Handler provide additional functions for maintaining object sets. Functions include transferring object sets between different Natural libraries and system files and/or different mainframe platforms and deleting or finding object sets in a different environment.

When using a Natural utility, an object set is treated like any other source object of the type Text.

For details, refer to the relevant sections in the *Utilities* documentation.

# Blacklist Maintenance in Batch Mode

SYSBPM blacklist maintenance functions can also be executed in batch mode as described in ***Batch Processing***.

In addition, the Natural system library SYSBPM provides the command `BPMBLBAT` that further facilitates batch processing of the blacklist maintenance functions add and delete all objects.

Log on to the library SYSBPM and execute the `BPMBLBAT` command using the syntax indicated in the diagram below. The symbols used in the diagram are explained in the section *System Command Syntax* in the *System Commands* documentation.

```
          { FUNC=LOCK, BPNAME=bp-name, (object-specification) }
BPMBLBAT  {                                                   }
          { FUNC=RLS, BPNAME=bp-name                          }
```

This section covers the following topics:

- Explanation of Syntax
- Examples of Input

## Explanation of Syntax

The keywords and the *`object-specification`* clause indicated in the `BPMBLBAT` syntax diagram above are described in the following section. ***Examples of Input*** demonstrate the use of the keywords.

| Syntax Item | Explanation |
|---|---|
| `FUNC=LOCK` | Adds object names to a blacklist as described in ***Adding Objects*** in the section *Maintain Blacklist*. |
| `FUNC=RLS` | Deletes a complete blacklist with all object names contained in the list. This functions corresponds to delete function described under ***To delete all objects from the blacklist*** in the section *Maintain Blacklist*. (In batch mode, you cannot delete single object names from a blacklist.) |
| `BPNAME` | The name of the buffer pool where the blacklist is loaded. |
| *`object-specification`* | The keywords that apply to the *`object-specification`* clause (see the respective syntax below) are: |
| | |
| | `LIB` — The name of the library where the objects are stored. |
| | |
| | `DBID` `FNR` — The database ID (DBID) and the file number (FNR) where the objects are stored. If DBID and FNR are left |

| Syntax Item | Explanation | |
|---|---|---|
| | | blank, they will be taken from the current system file FUSER or FNAT in libraries whose names start with SYS (except the library SYSTEM). |
| | | |
| | *object-name* | The name(s) of the object(s) to be added to the blacklist: Enter each name in a separate line in the positions 1 to 8.<br>To indicate the end of the input, in a separate line, enter a period (.) |

**Syntax of** *object-specification*

Shown and explained below is the syntax that applies to *object-specification*:

```
LIB=lib-name, DBID dbid, FNR fnr [object-name]...
```

## Examples of Input

### Example 1 - Adding Objects to a Blacklist

The example input below demonstrates how to add to a blacklist the objects A, B and C in the buffer pool NATGBP:

```
/*Job
.
.
.
*/Job

LOGON SYSBPM
BPMBLBAT
FUNC=LOCK,BPNAME=NATGBP,LIB=SAGTEST,DBID=10,FNR=32
A
B
C
.
FIN
```

### Example 2 - Deleting a Blacklist:

The example input below demonstrates how to delete a blacklist in the buffer pool NATGBP:

```
/*Job
.
.
.
*/Job
LOGON SYSBPM
BPMBLBAT
FUNC=RLS,BPNAME=NATGBP
.
FIN
```

# 65 Preload List Maintenance


- List Preload Lists ..... 412
- Edit Preload List ..... 413
- Generate Preload List ..... 417
- Delete Preload List ..... 419
- Additional Maintenance Functions with Utilities ..... 420

This function only applies to buffer pools of type Natural and the message pool.

**Preload List Maintenance** is used to maintain preload lists. In a preload list, you can specify the names of Natural objects or Natural messages that are to be loaded into the buffer pool or message pool when it is initialized.

The source code of a preload list is stored as a Natural source object of the type `Text` in the Natural system library SYSBPM. The first line of a preload list source contains the comment `**BPL**` (buffer pool preload list) or `**MPL**` (message pool preload list).

For further information on the preload list, see the relevant section in *Natural Buffer Pool* or *Message Pool* in the *Operations* documentation.

### To invoke Preload List Maintenance

- In the SYSBPM **Main Menu**, enter `B`, `C` or `*` to select a Natural buffer pool or `M` to select a message pool and the following function code:

  ```
  P
  ```

  Or:

  Enter one of the following SYSBPM direct commands:

  `PRELOADLIST` to invoke the menu for Natural objects

  or

  `MPRELOADLIST` to invoke the menu for Natural messages

  The **Preload List Maintenance** menu appears.

The functions provided in the **Preload List Maintenance** menu are explained in the following section. This section also provides information on additional preload list maintenance functions.

## List Preload Lists

This function invokes the **List Preload Lists** screen which displays a list of all existing preload lists.

### To invoke List Preload Lists

- In the **Preload List Maintenance** menu, enter function code `L` and the name of a preload list. Asterisk (*) notation is also allowed for a preload list name.

  Or:

Enter one of the following SYSBPM direct commands:

`LIST PRELOADLIST list-name` to list preload lists containing Natural objects

or

`LIST MPRELOADLIST list-name` to list preload lists containing Natural messages

Asterisk (*) notation is also allowed for `list-name`.

The **List Preload Lists** screen appears.

For a list of available line commands, enter a question mark (?) in the leftmost screen column (prefix information).

For a list of commands that can be entered in the command line of the **List Preload Lists** screen, invoke the **Help** window by entering a question mark (?) in the command line. If required, press PF7 to scroll backward and PF8 to scroll forward in the window.

## Edit Preload List

This function invokes the **Edit Preload List** screen where you can create a new preload list, add objects to an existing list or delete objects from it.

**Important:** The editing functions provided on the **Edit Preload List** screen are a subset of the functions provided by the Software AG Editor (described in the *Editors* documentation). Therefore, before you start a Natural session to edit a preload list, set the Natural profile parameter `EDPSIZE` to a value greater than `0` (zero); see also *Profile Parameters* in the *Parameter Reference* documentation. We recommend that you set `EDPSIZE` to a minimum of `100`.

For a list of available line commands, enter a question mark (?) in the leftmost screen column (prefix information).

For a list of commands that can be entered in the command line of the **Edit Preload List** screen, invoke the **Help** window by entering a question mark (?) in the command line. If required, press PF7 to scroll backward and PF8 to scroll forward in the window.

This section covers the following topics:


- Creating Preload Lists

- Modifying Preload Lists

## Creating Preload Lists

### To create a preload list

1 In the **Preload List Maintenance** menu, perform the following steps:

- Enter function code `E`.
- Clear the contents of the field **Preload List Name**, that is, do *not* enter the name of a preload list.
- In the fields **Library** and **Objects**, leave the default asterisk (*).

The **Edit Preload List** screen appears.

2
- **For Natural Objects**
  In the relevant input fields, enter the name of the library where the objects are stored, the names of the objects and the corresponding database IDs (**DBID**) and file numbers (**FNR**).

  If **DBID** and **FNR** are left unspecified, a default setting is provided by the current system file FUSER or FNAT in libraries whose names start with SYS (except the library SYSTEM).

  The resident flag will be set to `Y` (Yes) in column **R** on the editing screen if no value is entered. Resident means that the object is not deleted from the buffer pool, not even if its **Use Count** changes to `0` (zero). (**Use Count** corresponds to **Current Use Count** described in *Directory Information*.)
- **For Messages**
  In the relevant input fields, enter the name of the library where the user messages are stored (a blank for Natural system messages), the numbers of the messages (this field cannot be left unspecified), the corresponding database IDs (**DBID**) and file numbers (**FNR**), the code page and the language code.

  If **DBID** and **FNR** are left unspecified, default values are provided during the preload process by the current system file FUSER or FNAT in libraries whose names start with SYS (except the library SYSTEM). If no messages are found on FUSER or FNAT, search is extended to steplibs.

  If **Code Page** is left unspecified, a default code page is provided by the session code page during the preload process.

  If **Language Code** is left unspecified, a default language is provided by the system variable `*LANGUAGE` during the preload process.

3 To save the preload list as a source object of type Text in the library SYSBPM, enter the following command in the command line:

```
SA set-name
```

See also *Generate Preload List from Buffer Pool*.

### Modifying Preload Lists

#### To add a new object to a preload list

1 In the **Preload List Maintenance** menu, enter function code `E` and the name of a preload list.

   Or:

   On the **List Preload Lists** screen, in the leftmost column, next to the preload list required, enter the following line command:

```
E
```

   Or:

   Enter the following SYSBPM direct command:

```
EDIT PRELOADLIST list-name
```

   for a preload list containing Natural objects, or

```
EDIT MPRELOADLIST list-name
```

   for a preload list containing messages.

   The **Edit Preload List** screen appears with the specified preload list.

2 Complete the input fields as described in **step 2** in *Creating Preload Lists*.

3 Enter your modifications in preload list you have selected.

4 To save the modifications, in the command line of the **Edit Preload List** screen, enter the following:

```
SA
```

#### To modify an object of a preload list

1 In the **Preload List Maintenance** menu, enter function code `E`, a library name and the name of a preload list.

   Or:

   On the **List Preload Lists** screen, in the leftmost column, next to the preload list required, enter the following line command:

```
E
```

Or:

Enter the following SYSBPM direct command:

```
EDIT PRELOADLIST list-name
```

for a preload list containing Natural objects, or

```
EDIT MPRELOADLIST list-name
```

for a preload list containing messages.

Or:

The **Edit Preload List** screen appears with the specified preload list.

2 In the relevant input field(s), replace the existing entries with new values. The same rules are applied as described in **step 2** in *Creating Preload Lists*.

3 To save the modifications, in the command line of the **Edit Preload List** screen, enter the following:

```
SA
```

### To delete an object from a preload list

1 In the **Preload List Maintenance** menu, enter function code `E`, a library name and the name of a preload list.

Or:

On the **List Preload Lists** screen, in the leftmost column, next to the preload list(s) required, enter the following line command:

```
E
```

Or:

Enter the following SYSBPM direct command:

```
EDIT PRELOADLIST list-name
```

for a preload list containing Natural objects, or

```
EDIT MPRELOADLIST list-name
```

for a preload list containing messages.

The **Edit Preload List** screen appears with the specified preload list.

2 In the leftmost column, next to the object required, enter the following line command:

```
D
```

The specified object is removed from the preload list.

3 To save the modification, in the command line of the **Edit Preload List** screen, enter the following:

```
SA
```

## Generate Preload List

- Generate Preload List from Buffer Pool
- Generate Preload List from Message Pool

### Generate Preload List from Buffer Pool

This function is used to generate a new preload list by using the names of the objects currently loaded in the buffer pool. From the objects that are currently in the buffer pool, you can select those you wish to be included in the preload list.

**To generate a preload list, use either of the following options:**

1 In the **Preload List Maintenance** menu, enter function code `G` and the name of a preload list. In the fields **Library**, **Objects**, **Resident**, **Use Count** and **Total Use Count**, specify the objects to be included in the list:

- To include all objects that are currently in the buffer pool, enter an asterisk (*) in the fields **Library**, **Objects** and **Resident**, and leave the fields **Use Count** and **Total Use Count** blank.
- Or:
  To include specified objects in the buffer pool, in the fields described below, you can enter the following values:

| | |
|---|---|
| **Library** | A single name or asterisk (*) notation. |
| **Objects** | A single name or asterisk (*) notation. |
| **Resident** | An asterisk (*) for all objects or `Y` (Yes) for all objects currently marked as resident in the buffer pool. |
| **Use Count** | A numeric start *`value`* (>), for example `>10`. Selects all objects with a **Use Count** greater than or equal to *`value`*.<br><br>This field corresponds to **Current Use Count** described in *Directory Information*. |

| | |
|---|---|
| **Total Use Count** | A numeric start *value* (>), for example >10. Selects all objects with a **Total Use Count** greater than or equal to *value*.<br><br>This field corresponds to **BP Total Use** described in *Directory Information*. |

2 Enter either of the following SYSBPM direct commands:

```
GENERATE PRELOADLIST list-name
```

or

```
GENERATE PRELOADLIST list-name gen-library
```

(See also the explanations of field values above).

A message appears confirming that the preload list was generated from the buffer pool.

All preload list objects will be generated as resident (entry Y in column **R**) by default. Choose manually, which objects you want to remove from the list.

Objects from the library SYSBPM will not be included in the generated preload list as it can be assumed that these are objects which were only loaded into the buffer pool in order to execute this function.

### Generate Preload List from Message Pool

This function is used to generate a new preload list by using the names of the objects currently loaded in the message pool. From the objects that are currently in the message pool, you can select those you wish to be included in the preload list.

**To generate a preload list, use either of the following options:**

1 In the **Preload List Maintenance** menu, enter function code G and the name of a preload list. In the fields **Library**, **Message**, **DBID**, **FNR**, **Language Code** and **Code Page**, specify the messages to be included in the list:

- To include all messages that are currently in the message pool, enter an asterisk (*) in the **Library** field, enter 1 - 9999 in the Message Number field, 0 in the fields **DBID**, **FNR** and **Language Code**, and leave the field **Code Page** unspecified.
- Or:
  To include specified messages in the message pool, in the fields described below, you can enter the following values:

| | |
|---|---|
| **Message Number** | Two numbers between 1 and 9999, in ascending order. |
| **Library** | A single name or asterisk (*) notation. |
| **DBID** | A numeric value between 0 and 65535, where 0 means all messages. |
| **FNR** | A numeric value between 0 and 5000, where 0 means all messages. |
| **Language Code** | A numeric value between 1 and 60. |
| **Code Page** | A single name or blank for all values. |

2 Enter either of the following SYSBPM direct commands:

```
GENERATE MPRELOADLIST list-name
```

or

```
GENERATE MPRELOADLIST  list-name start-number end-number gen-library language ↩
dbid fnr codepage
```

(See also the explanations of field values above).

A message appears confirming that the preload list was generated from the message pool.

## Delete Preload List

### To delete a preload list

1 In the **Preload List Maintenance** menu, enter function code `L` and the name of a preload list.

Or:

Enter the following SYSBPM direct command:

```
LIST PRELOADLIST list-name
```

for a preload list containing Natural object, or

```
LIST MPRELOADLIST list-name
```

for a preload list containing messages.

Or:

```
LIST MPRELOADLIST list-name
```

The **List Preload Lists** screen appears.

2 In the leftmost column, next to the object required, enter the following line command:

```
D
```

The **DELETE** window appears.

3 Confirm the deletion by entering the name of the preload list.

A confirmation message appears.

## Additional Maintenance Functions with Utilities

The Natural utilities SYSMAIN and Object Handler provide additional functions for maintaining preload lists. Functions include transferring preload lists between different Natural libraries and system files and/or different mainframe platforms and deleting or finding preload lists in a different environment.

When using a Natural utility, a preload lists is treated like any other source object of the type Text.

For details, refer to the relevant sections in the *Utilities* documentation.

# 66 Performance Considerations


- Internal Fast Locate Table ..... 422
- Searching in Steplibs ..... 423
- Reusing and Retaining Objects ..... 423
- Local versus Global Buffer Pool ..... 424

This section provides general advice on performance-related issues regarding the buffer pool and the BP cache.

For explanations of the statistics items mentioned in this section, refer to ***Buffer Pool Load/Locate Statistics***.

**Related Topic:**

- ***Performance Hints***

## Internal Fast Locate Table

When a Natural object is referenced for the first time within a Natural session, the buffer pool manager creates a directory entry for this object. A directory entry is used to identify an object and contains information such as the name of the object, the library where it resides (name, database ID and file number) and the address of the object (position) in the buffer pool.

The Natural runtime system remembers names of objects that were recently executed, the libraries (name, database ID and file number) where they reside and the addresses of the corresponding buffer pool directory entries in the internal fast locate table for the duration of a Natural session.

When a user invokes an object that has been used before in the Natural session, the Natural runtime system passes the information from the internal fast locate table to the buffer pool manager, which can then bypass the Normal Locate procedure and perform a time-saving Quick Locate call (see **Quick Locate Calls**). This is the most efficient way to locate an object.

If the position of an object in the buffer pool has changed, the buffer pool manager will automatically schedule a Normal Locate call. The position usually changes, if an object that was deleted from the buffer pool or overwritten by another object is reloaded into the buffer pool.

The internal fast locate table contains a maximum of 128 entries. The fast locate table is reset with the `LOGON` system command and must be filled again using Normal Locate calls. Therefore, an application that performs a **LOGON** loses performance.

A high ratio of **Normal Locate Calls** to **Quick Locate Calls** indicates the use of `LOGON` commands in a Natural application in that each `LOGON` causes the internal fast locate table to be reset.

## Searching in Steplibs

The search for a Natural object through a long chain of steplib libraries has a negative impact on performance.

For each steplib, the Natural runtime issues a call to the buffer pool manager until the requested object is found. However, each needless call for a steplib library that does not contain the requested object can usually be avoided.

Depending on the setting of the `BPSFI` profile parameter (described in the *Parameter Reference* documentation), additional database calls may be required.

A long steplib chain is indicated by a high ratio of **Normal Locate Calls** (including steplib searches) to **Normal Locate Calls** (without steplib searches) that is calculated as follows:

**Normal Locate Calls: (Normal Locate Callslls - STEPLIB Searches)**

### Example of a Steplib Search

When searching the default steplib chain (library SYSTEM of FUSER, library SYSTEM of FNAT), each attempt to load an object from SYSTEM (FNAT) will result in the following:

3 **Normal Locate Calls** and 2 **STEPLIB Searches**

Explanation:
3 Normal Locate calls result from searches in the current library, the library SYSTEM (FUSER) and the library SYSTEM (FNAT). There are (at least) 2 Normal Locate calls that fail, since the object is stored in neither the current library nor the library SYSTEM (FUSER). Using the above formula, results in a ratio of 3:1.

If the object is located in the current library, the result is as follows:

1 Normal Locate call and 0 (zero) **STEPLIB Searches**. Using the above formula, results in a ratio of 1:1.

## Reusing and Retaining Objects

An application that contains many Natural objects where each object is rarely ever executed has heavy impact on the performance of the buffer pool. None of the objects resides in the buffer pool for a long time and many objects have to be loaded from a system file. For performance reasons, an application should reuse objects as often as possible, for example, by moving identical source code contained in multiple objects into a single object.

You can check the use of objects with the **List Objects** function (see the section *List Objects*). For example: the **Max** column contains information on the maximum number of applications that have executed an object, and the **TotalUC** column contains the total number of Locate calls of an object that was loaded into the buffer pool.

Objects can be made resident individually by using the **List Objects** function or by specifying them in a preload list (see ***Preload List Maintenance***).

## Local versus Global Buffer Pool

There are no general recommendations for when to use a local or a global buffer pool as different applications have different requirements. However, rules-of-thumb from experience enable us to give general advice on:

This section covers the following topics:

- Using Local Buffer Pools
- Using a Global Buffer Pool

### Using Local Buffer Pools

Performance can improve using several small local buffer pools instead of a single global buffer pool if local buffer pools can be assigned to different application environments.

For example, in CICS environments, performance usually improves when using a local buffer pool for each AOR (Application Operating Region).

### Using a Global Buffer Pool

For different batch applications that reference the same Natural objects, performance may improve if these applications use a common global buffer pool instead of a local buffer pool for each individual application. If so, there is a higher probability that the objects required by each application have already been loaded into the global buffer pool by one of the other applications.

# 67 SYSBPM Direct Commands

The SYSBPM direct commands described in this section can be used to directly execute SYSBPM utility functions or navigate in SYSBPM screens in online or batch mode. For additional SYSBPM direct commands that only apply to particular screens, refer to the sections where the individual SYSBPM functions are documented.

SYSBPM direct commands that refer to the BP cache or buffer pool hash table only apply to buffuffer pools of the type Natural (`TYPE=NAT`), see *TYPE - Type of Buffer Pool* in the *BPI - Buffer Pool Initialization* documentation.

The following table lists all SYSBPM direct commands (including subcommands) provided, the parameters that can be used with the commands, and the equivalent SYSBPM menu functions.

The SYSBPM direct commands listed below can be entered in the Command line of any SYSBPM screen. To execute a SYSBPM direct command from any other command prompt or in batch mode, the direct command must be preceded by the keyword SYSBPM, for example:

```
SYSBPM ADD BLACKLIST
```

An underlined portion of a SYSBPM command represents its minimum abbreviation. Parameter values that are required by a command are represented by letters in italics.

| Command | Parameters | Function |
|---|---|---|
| + | none | Scrolls one page down in a list. |
| - | none | Scrolls one page up in a list. |
| ADD BLACKLIST | none | Invokes the **Maintain Blacklist** screen described in *Blacklist Maintenance*. |
| ADD SET | *library-name set-name* | Adds all objects of a specified object set to the blacklist as described in ***Add Object Set to Blacklist***. |
| BLACKLIST | none | Invokes the **Blacklist Maintenance** menu. |
| BOTTOM | none | Scrolls to the end of a list. |

| Command | Parameters | Function |
|---|---|---|
| CANCEL | none | Same as EXIT. |
| CHECK HASH<br>or<br>CHECK HT | none | Checks the BP hash table for consistency and returns the number of inconsistencies found.<br>See also REBUILD HASH. |
| CLOSE BPC | none | BP cache required.<br>Closes the BP cache. The buffer pool runs without BP cache afterwards. You can restart the BP cache by using the INITIALIZE BPC command. |
| DELETE | none | Deletes all objects from the buffer pool and the BP cache.<br>If entered on the **Directory Information** screen: see DELETE in ***PF Keys and Direct Commands***. |
| DELETE | *library-name object-name dbid fnr* | Deletes the specified object(s) from the buffer pool and the BP cache as described in ***Delete Objects***. |
| DELETE ALL | none | Deletes all objects from the blacklist as described in ***Delete Object Set from Blacklist***. |
| DELETE BUFFERPOOL<br>or<br>DELETE BP | none | Deletes all objects from the buffer pool only. |
| | *library-name object-name dbid fnr* | Deletes the specified object(s) from the buffer pool only. |
| DELETE BPC | none | BP cache required.<br>Deletes all objects from the BP cache only. |
| | *library-name object-name dbid fnr* | BP cache required.<br>Deletes the specified object(s) from the BP cache only. |
| DELETE BLACKLIST | none | Invokes the **Maintain Blacklist** screen where you can delete blacklist entries as described in *Blacklist Maintenance*. |
| DELETE MP | none | Message pool required.<br>Deletes all messages from the message pool. |
| | *start-number end-number library-name language dbid fnr code-page* | Message pool required.<br>Deletes the specified messages from the message pool. |
| DELETE SET | *library-name set name* | Deletes all objects of the specified object set from the blacklist as described in ***Delete Object Set from Blacklist*** |

| Command | Parameters | Function |
|---|---|---|
| *library-name object-name dbid fnr* | Deletes the specified object(s) from the buffer pool only. | . |
| DISPLAY ALL | none | Same as DISPLAY LIST. |
| DISPLAY BUFFERPOOL<br><br>or<br><br>DISPLAY BP | none | See ***Display Buffer Pools*** in *Select Buffer Pool*. |
| DISPLAY BLACKLIST | none | Invokes the **Maintain Blacklist** screen described in *Blacklist Maintenance*. |
| DISPLAY CDIRECTORY | none | BP cache required.<br><br>Invokes the **Directory Information** screen. |
| DISPLAY CGENERAL | none | BP cache required.<br><br>Invokes the **General BP Cache Statistics** screen described in *General BP Cache Statistics*. |
| DISPLAY CHASH | none | Invokes the function **BP Cache Hash Table Statistics** described in *BP Cache Hash Table Statistics* and displays the **Cache Hash Table Collisions** screen. |
| DISPLAY CLIST | *library-name object-name dbid fnr* | BP cache required.<br><br>Invokes the **List Objects** screen.<br><br>In contrast to the command DISPLAY LIST, this command generates a statistics report that displays data about BP cache objects at the beginning of the list. |
| DISPLAY CLOAD | none | BP cache required.<br><br>Invokes the **BP Cache Call Statistics** screen described in *BP Cache Statistics*. |
| DISPLAY CSTATISTICS | none | BP cache required.<br><br>Invokes the **BP Cache Statistics** menu described in ***BP Cache Statistics***. |
| DISPLAY DIRECTORY | *library-name object-name dbid fnr* | Invokes the **Directory Information** screen. |
| DISPLAY FRAGMENTATION | none | Invokes the **Buffer Pool Fragmentation** screen described in *Buffer Pool Statistics*. |
| DISPLAY FUNCTION | none | Invokes the **Internal Function Usage** screen described in *Buffer Pool Statistics*. |
| DISPLAY GENERAL | none | Invokes the **General Buffer Pool Statistics** screen described in *Buffer Pool Statistics*. |

| Command | Parameters | Function |
|---|---|---|
| DISPLAY HASH<br><br>or<br><br>DISPLAY HT | none | Invokes the function **Buffer Pool Hash Table Statistics** and displays the **Hash Table Collisions** screen. |
| DISPLAY HDIRECTORY | *library-name object-name dbid fnr* | Invokes the **Directory Information Hex** screen that displays in hexadecimal format the directory information of an object. |
| DISPLAY HEX | *library-name object-name dbid fnr* | Invokes the **Hexadecimal Display** screen that displays in hexadecimal format the source of an object. |
| DISPLAY LIST | *library-name object-name dbid fnr* | Invokes the **List Objects** screen.<br><br>In contrast to the command DISPLAY CLIST, this command generates a statistics report that displays data about buffer pool objects at the beginning of the list. |
| DISPLAY LOAD | none | Invokes the **Buffer Pool Load/Locate Statistics** screen described in *Buffer Pool Statistics*. |
| DISPLAY MLIST | Start-number end-number library-name language dbid fnr code-page | Message pool required.<br><br>Invokes the **List Objects** screen. Lists the selected messages from the message pool. |
| DISPLAY MP | none | Message pool required.<br><br>List all available message pools running in the current subsystem. |
| DISPLAY MSTATISTICS | none | Message pool required.<br><br>Displays **Message Pool Statistics** . |
| DISPLAY PERFORMANCE | none | Invokes the **Performance Hints** screen with performance-related statistics of a Natural buffer pool and BP cache as described in *Buffer Pool Statistics* and *BP Cache Statistics*. |
| DISPLAY STATISTICS | none | Invokes the **Buffer Pool Statistics** menu described in *Buffer Pool Statistics*. |
| EDIT PRELOADLIST | *list-name* | Invokes the **Edit Preload List** screen described in *Preload List Maintenance*. |
| EDIT MPRELOADLIST | *list-name* | Message pool required.<br><br>Invokes the **Edit Message Preload List** screen described in *Preload List Maintenance*<br>. |
| EDIT SET | *library-name set-name* | Invokes the **Edit Object Set** screen described in *Blacklist Maintenance*. |
| EXIT | none | Exit the current screen and displays the previous screen. |

| Command | Parameters | Function |
|---|---|---|
| FLIP | none | Switches the PF-key line: toggles between the display of PF1 to PF12 and PF13 to PF24. |
| GENERATE MPRELOADLIST | *list-name start-number end-number gen-library language dbid fnr code-page* | Message pool required.<br><br>Invokes the function **Generate Preload List from Message Pool**. |
| GENERATE PRELOADLIST | *list-name gen-library* | Invokes the function **Generate Preload List from Buffer Pool**. |
| INITIALIZE | none<br><br>or:<br>1, 2, 4, 8, 12, 16 | Reinitializes the buffer pool and the BP cache. If no text record size is specified, the current text record size will be taken.<br><br>Only use this function if the **Current Use Count** (see *Directory Information*) is equal to 0 (see the warning below) or if the buffer pool has been destroyed.<br><br>**Caution:** If you try to reinitialize the buffer pool while objects are being executed by active sessions in this buffer pool, the **Confirm Initialization** window appears with the **Current Use Count** for this buffer pool (not counting the SYSBPM user himself). If **Current Use Count** is *not* equal to 0 (zero) and you enter a Y to confirm the reinitialization of the buffer, the results of the active sessions are unpredictable and Natural can even abend. |
| INITIALIZE BP | none<br><br>or:<br>1, 2, 4, 8, 12, 16 | Reinitializes the buffer pool only. If no text record size is specified, the current text record size will be used.<br><br>See also the ***Warning*** above. |
| INITIALIZE BPC | none | BP cache required.<br><br>Reinitializes the BP cache only. The text record size of the BP cache is fixed (4 KB). |
| INITIALIZE MP | none | Message pool required.<br><br>Reinitializes the message pool active for SYSBPM, i.e. the contents of the message pool will be deleted. |
| LAST | none | Displays the SYSBPM direct command entered most recently. |
| LIST MPRELOADLIST | *list-name* | Message pool required.<br><br>Invokes the **Generate Preload List from Message Pool** screen for the specified object as described in ***Preload List Maintenance***. |

| Command | Parameters | Function |
|---|---|---|
| LIST PRELOADLIST | *list-name* | Invokes the **List Preload Lists** screen for the specified object as described in *Preload List Maintenance*. |
| LIST SET | *library-name*<br>*set-name* | Invokes the **List Object Sets** screen for the specified library or object as described in *Blacklist Maintenance*.<br><br>Asterisk (*) is also allowed for *set-name*. |
| MENU | none | Invokes the SYSBPM **Main Menu** as described in ***Invoking and Operating SYSBPM***. |
| MPRELOADLIST | none | Message pool required.<br><br>Invokes the **Preload List Maintenance** menu described in *Preload List Maintenance*. |
| PRELOADLIST | none | Invokes the **Preload List Maintenance** menu described in *Preload List Maintenance*. |
| QUIT | none | Same as EXIT. |
| REBUILD HASH<br><br>or<br><br>REBUILD HT | none | Rebuilds hash tables if inconsistencies are found with CHECK HASH.<br><br>REBUILD HASH deletes the current hash table and rebuilds a new hash table from the current buffer pool contents. |
| RESET BUFFERPOOL<br><br>or<br><br>RESET BP | none | Resets the buffer pool as described in ***Reset Buffer Pool***. |
| RESET MP | none | Resets the message pool active for SYSBPM to the message pool used when Natural is started. |
| SELECT BUFFERPOOL<br><br>or<br><br>SELECT BP | none<br>or:<br>*buffer pool name* | Only applies to buffer pools of the type Natural, DL/I or Sort.<br><br>Invokes a selection list of buffer pools as described in ***Select Buffer Pool***.<br><br>If a buffer pool name is entered, it will be selected without showing the selection list. |
| SELECT MP | none<br>or:<br>*message pool name* | Message pool required.<br><br>Invokes a selection list of message pools as described in ***Select Buffer Pool***.<br><br>If a message pool name is entered, it will be selected without showing the selection list. |
| SORT BPC | See ***Command Syntax for SORT BPC*** below. | Sorts the BP cache as described in ***Display Sorted Extract***. |

| Command | Parameters | Function |
|---|---|---|
| SORT BUFFERPOOL | See *Command Syntax for SORT* below. | Sorts the buffer pool as described in *Display Sorted Extract*. |
| STOP | none | Leaves the SYSBPM utility. |
| TOP | none | Scrolls to the beginning of a list. |
| WRITE BP<br>or<br>WRITE BPC<br>or<br>WRITE ALL | | Writes object directory data to a local file or a PC text file.<br>See also *Write to Work File*. |

### Command Syntax for SORT

```
SORT [ { BUFFERPOOL } ] { OBJECT } [ { D } ]
       { BP       }     { TOTAL  }   { A }
                        { LAST   }
```

### Command Syntax for SORT BPC

```
SORT BPC { OBJECT } [ { D } ]
         { TOTAL  }   { A }
         { LAST   }
```

# 68 Batch Processing


- Related Topics ..... 434

The functions provided by the SYSBPM utility can also be executed in batch mode.

For this purpose, we recommend that you use the SYSBPM Application Programming Interface USR4340N or USR0340N described in the section *Application Programming Interfaces*.

You can also use the SYSBPM utility in batch by simulating the online input command sequence.

Since SYSBPM uses functionality of the Software AG Editor, when writing your batch job, observe the following instructions:

- Set the profile parameter `EDPSIZE`. As an alternative, set the profile parameter `BPI` with `TYPE=EDIT` for the Software AG Editor buffer pool and define the editor work file in the batch job
- Note that the field **Code** is not available on every SYSBPM screen.
- Note that you may have to skip input fields in order to position the cursor in the Command line input field for entering direct commands.
- Simulate PF keys by using the terminal commands `%K` to navigate through the SYSBPM utility. For example, use `%K3` to leave the **List Objects** screen.
- Use the continuation character defined with the session parameter `CF` (default is `%`) to enter more than one line of input for a map.

Listed below are the topics and Natural documentation sources that refer to the instructions mentioned above.

## Related Topics

| Topic | Documentation |
|---|---|
| *Blacklist Maintenance in Batch Mode* | *SYSBPM Utility* |
| *SYSBPM Direct Commands* | *SYSBPM Utility* |
| *EDPSIZE - Size of Software AG Editor Auxiliary Buffer Pool* | *Parameter Reference* |
| *BPI - Buffer Pool Initialization* | *Parameter Reference* |
| *Installing the Software AG Editor* | *Installation for z/OS*<br>*Installation for z/VSE*<br>*Installation for BS2000* |
| *%K and %KP - Simulate PF- and PA-Key* | *Terminal Commands* |
| *EDBP - Software AG Editor Buffer Pool Definitions* | *Parameter Reference* |
| *Editor Buffer Pool* | *Operations* |
| *Editor Work File* | *Operations* |
| *Natural in Batch Mode* | *Operations* |
| *Using the INPUT Statement in Non-Screen Modes* | *Statements* |
| *Using the INPUT Statement in Batch Mode* | *Statements* |

# 69 Application Programming Interfaces

This section describes the Application Programming Interfaces (APIs) USR0340N, USR0341N, USR4340N, USR4341N and USR4342N, which are used for handling Natural objects currently loaded in the buffer pool and/or BP cache. The APIs are supplied in the Natural system library SYSEXT.

For further information on these APIs, see the following:

- The relevant Natural source object of the type Text and the example programs in the Natural system library SYSEXT.
- The SYSBPM functions referenced in the table below.

**Related Topic:**

- *SYSBPM Batch Processing*

| API | Functionality |
|---|---|
| USR0340N | ■ Deletes objects from the buffer pool and/or BP cache.<br>■ Marks objects as resident.<br>■ Removes the resident flag from objects.<br>■ Reads object directory information.<br>■ Retrieves general buffer pool statistics and buffer pool load/locate statistics.<br><br>Corresponding SYSBPM functions:<br><br>**Delete Objects**<br>**List Objects**<br>**Directory Information**<br>**General Buffer Pool Statistics**<br>**Buffer Pool Load/Locate Statistics** |

| API | Functionality |
|---|---|
| USR0341N | Collects garbage to clean up the buffer pool by removing objects which are no longer needed.<br><br>Selection criteria for specified objects are the relative age of an object. Relative age is the time the object has been loaded in the buffer pool which calculates from **BP Last Action** date. Minimum age is 30 minutes.<br><br>See also the SYSBPM function **List Objects**. |
| USR4340N | We recommend that you use this interface for batch processing instead of using the SYSBPM utility in batch.<br><br>Lists objects loaded in the buffer pool and/or BP cache sorted by **Object Size** (ObjSize), **Total Use Count** (TotalUC) or **BP Last Action**. **BP Last Action** only applies to the buffer pool.<br><br>Corresponding SYSBPM functions:<br><br>**Display Sorted Extract**<br>**List Objects** |
| USR4341N | Maintains a blacklist of Natural objects, which are not to be executed and loaded into the buffer pool.<br><br>See also the SYSBPM function **Blacklist Maintenance**. |
| USR4342N | ■ Deletes objects from the message pool.<br>■ Lists messages from the message pool.<br>■ Retrieves message pool statistics.<br><br>Corresponding SYSBPM functions:<br><br>**Delete Objects**<br>**List Objects**<br>**Message Pool Statistics** |

# XII SYSCP Utility - Code Page Administration

# 70 SYSCP Utility - Code Page Administration


- Invoking and Terminating SYSCP ..... 440
- Code Page Maintenance ..... 442
- All Code Pages ..... 454
- Unicode Properties ..... 460
- ICU Information ..... 461

The SYSCP utility is used to obtain information on code pages and ICU (International Components for Unicode) data libraries available in the current Natural mainframe environment. In addition, you can use the SYSCP utility to change the code page assignment of programming objects, DDMs, profiles and error messages. You can also add, remove or convert code pages.

The SYSCP utility also offers you help to avoid problems that can occur when a code page is not defined or enabled in Natural or when objects are converted to an incorrect code page or Unicode format.

For detailed information on how Natural supports Unicode and code pages and Unicode-specific items, see the descriptions and presentations in the *SYSEXV Utility* and ***Related Topics*** below.

**Note:** The use of the SYSCP utility can be controlled by Natural Security. For detailed information, see the section *SYSCP - Code Page Administration - Utility Profiles* in the *Natural Security* documentation.

**Related Topics:**

- *Unicode and Code Page Support*: *Natural* documentation
- Unicode: Unicode Consortium at web site at ***http://www.unicode.org/***
- ICU: International Components for Unicode at web site ***http://site.icu-project.org/***
- ICU: Converter Explorer documentation at web site ***http://demo.icu-project.org/icu-bin/convexp***

## Invoking and Terminating SYSCP

Instructions for invoking and terminating the SYSCP utility and performing a function are provided in the following section.

### To invoke the SYSCP utility

- Enter the following system command:

```
SYSCP
```

A SYSCP menu similar to the example below appears:

```
11:19:07              ***** NATURAL SYSCP UTILITY *****            2010-02-18
User SAG                            - Menu -              ICU Version 4.0.1
                                                        Unicode Version 5.1


                          Function

                       _  Code Page Maintenance
                       _  All Code Pages
                       _  Unicode Properties
                       _  ICU Information
                       _  Help
                       _  Exit





Command ===>




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

The current ICU and Unicode versions are indicated at the top of the screen.

The functions contained in the menu are explained in the remainder of this documentation.

### To execute a SYSCP function

- In the SYSCP menu, place the cursor in the input field next to the required function and press ENTER.

  Or:

  In the SYSCP menu, in the input field next to the required function, enter any character and press ENTER.

**Note:** In the Command line of any SYSCP utility screen, you can enter any Natural system command. A system command terminates the SYSCP utility.

### To terminate SYSCP

- Press PF3 or PF12.

Or:

From the SYSCP menu, choose **Exit**.

## Code Page Maintenance

The **Code Page Maintenance** menu provides functions that can be applied to programming objects (N), DDMs (D), profiles (P), user error messages (E) and system error messages (S). For an object you can list (L), assign (A) or remove (R) code page information. You can also convert (C) a code page or check its conversion for assigned (K) or unassigned (C) objects.

All code page maintenance functions reference the standard IANA name (see also **Cmd** in *All Code Pages*); you cannot use a code page name other than IANA when you execute a code page maintenance function.

The results of a code page maintenance function are output on a report screen, which is described in ***Function Result Report***.

When you invoke **Code Page Maintenance**, a maintenance menu similar to the example below appears:

```
16:53:05                ***** NATURAL SYSCP UTILITY *****             2012-06-20
User SAG                    - Code Page Maintenance -

        Code Function                                 Code Object

          L  List Code Page Information                 N  Programming Objects
          C  Check Conversion of Unassigned Objects     D  DDMs
          A  Assign Code Page Information               P  Profiles
          K  Check Conversion of Assigned Objects       E  User Error Messages
          T  Convert to Different Code Page             S  System Error Messages
          R  Remove Code Page Information
          U  Convert to Unshaped Form
          ?  Help
          .  Exit

 Function _                                   Object .. N

 Library  SSZ_____      DBID ...... _____     FNR ..... _____
                        Password ..           Cipher ..


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

The fields and functions contained in the **Code Page Maintenance** menu and the options and features provided by the functions are explained in the following sections:


- Code Page Maintenance Menu
- List Code Page Information
- Check Conversion of Unassigned Objects
- Assign Code Page Information to Objects
- Check Conversion of Assigned Objects
- Convert to Different Code Page
- Remove Code Page Information from Sources
- Convert to Unshaped Form
- Name Specification
- Object Selection List
- Function Result Report


## Code Page Maintenance Menu

The fields contained in the **Code Page Maintenance** menu are explained in the following table:

| Field | Explanation |
|---|---|
| **Function** | The code for the function to be executed. |
| **Object** | The code for the type of object the function is applied to. |
| **Library** | The name of the Natural library that contains the objects for which to execute a code page maintenance function.<br><br>The name entered by default is the name of the current library as specified with the system variable `*LIBRARY-ID` (see the *System Variables* documentation).<br><br>Not used for DDMs, profiles, and system error messages. |
| **DBID** | The database ID (DBID) of the Natural system file where the specified library is stored.<br><br>If no value (or `0`) is specified, the corresponding system file is used. |
| **FNR** | The file number (FNR) of the Natural system file where the specified Natural library is stored.<br><br>If no value (or `0`) is specified, the file number of the corresponding system file is used. |
| **Password** | If the specified system file is password protected, you must supply the appropriate 8-character Adabas password. |
| **Cipher** | If the specified system file is enciphered, you must supply the appropriate 8-digit Adabas cipher code. |

### List Code Page Information

To list code page information for objects of a specific kind, enter `L` in field **Function** and the code specific to these objects in field **Object**. For example, entering `P` results in a screen that lists all profiles, and the respective code page if a code page has been assigned, see the example below.

```
09:52:02                ***** NATURAL SYSCP UTILITY *****            2011-03-22
User SAG           - List Code Page Information of Profiles -
                                                        Listed Library SAG
Cmd  Name      Code Page                          Type
---  *________ *________________________________ *__
__   USER1                                        P
__   USER2                                        P
__   USER3     IBM01140                           P
__   USER4     IBM01140                           P
__   USER5                                        P
__   USER6                                        P
__   USER7                                        P
__   USER8     IBM01140                           P
__   USER9                                        P
__   USER10                                       P
__   USER11                                       P
__   USER12                                       P
__   USER13                                       P
__   USER14                                       P


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

The fields and columns contained in a **List Code Page Information** screen are explained in the following table. Note that some fields or columns are specific to the kind of object selected:

| Field/Column | Explanation | Object |
|---|---|---|
| **Listed Library** | See **Library** in *Code Page Maintenance Menu*. | N, E, P |
| **Cmd** | Input field for the following line command to be executed for a selected programming object:<br><br>`LD` Display object directory information.<br><br>This line command corresponds to the command `LIST DIRECTORY` *`object-name`* described in *Displaying Directory Information* in the *System Commands* documentation. | all objects |
| **Name** | The name of the object. | all objects, except E, S |
| **Code Page** | The code page information (IANA name) of the object. This column appears empty for an object that is not assigned a code page. | all objects |

| Field/Column | Explanation | Object |
|---|---|---|
| **Type** | The type of the object listed. | N, P |
| **Error Text** | The truncated text of the message. | E, S |
| **Numbr** | The number of the message. | E, S |
| **LC** | The language code character for the object, see `*LANGUAGE`. | E, S |
| **S/L** | Indication whether the message is a short message or a long message. | E, S |

### Filtering Objects

You can shorten the list of objects displayed on the **List Code Page Information** screen by specifying selection criteria.

#### To specify selection criteria

1 You can specify selection criteria for each column headed by an input field. Replace the default asterisk (*) with any of the input values listed in *Name Specification*.

2 In the input field underneath the column heading **Type**, replace the default asterisk (*) with one or more (maximum is 11) of the following type codes without a separator character:

| Code | Object Type | Code | Object Type |
|---|---|---|---|
| P | Program | A | Parameter data area |
| N | Subprogram | G | Global data area |
| S | Subroutine | L | Global data area |
| M | Map | C | Copycode |
| H | Helproutine | T | Text |
| 7 | Function | 4 | Class |
| 3 | Dialog | | |
| 5 | Processor | | |
| * | All Types | | |

### Check Conversion of Unassigned Objects

This function is used to check whether an unassigned object can be converted to a code page.

An unassigned object is an object without code page information, which was originally saved under a Natural version where code page information was not yet supported. Since no code page information is provided, you need to decide which code page to specify for the object to be checked for conversion. This depends on the character set used in the object.

For example, if you apply the function **Check Conversion of Unassigned Objects** to **Profiles**, a screen similar to the example below appears:

```
07:55:03                ***** NATURAL SYSCP UTILITY *****             2011-04-18
User SAG          - Check Conversion of Unassigned Profiles -


   Check if profiles that have no code page information can be
   converted from a given code page to a target code page.

   Use selection list .. Y

   Source code page .... IBM01140_____________________________________________

   Target code page .... IBM01140_____________________________________________

   Profile name ........ *________     DBID ...... 10___     FNR ..... 32___
                                       Password ..           Cipher ..
   Profile types ....... *__




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

The fields contained in the **Check Conversion of Unassigned Objects** screen are explained in the following table:

| Field | Explanation | Object |
|---|---|---|
| **Use selection list** | Specifies whether selective processing or automated processing is used for the specified objects: see *Object Selection List*. | all objects |
| **Source code page** | The name of the code page to be used to check whether the specified objects (to which no code pages are yet assigned) can be converted from this code page to the code page entered in the **Target code page** field. If the conversion check is successful, the code page specified in **Target code page** can be used with the **Assign Code Page Information to Objects** function.<br><br>The default name entered is the IANA name as returned by the `*CODEPAGE` system variable (see the *System Variables* documentation). | all objects |
| **Target code page** | The name of the code page to check for conversion of the specified unassigned objects.<br><br>The default name entered is the IANA name as returned by the `*CODEPAGE` system variable (see the *System Variables* documentation). | all objects |
| **Object name/Object name range** | The name of a single object or a range of names to be processed: see *Name Specification* for valid input values. | all objects |
| | The language code character for the object, see `*LANGUAGE` | E, S |

| Field | Explanation | Object |
|---|---|---|
| **Short/long error msgs** | Type of error message to be processed:<br><br>S process short error messages<br>L process long error messages<br>*/A process any error message | E, S |
| **Library** | See **Library** in *Code Page Maintenance Menu*. | N, E |
| **DBID** | See **DBID** in *Code Page Maintenance Menu*. | all objects |
| **FNR** | See **FNR** in *Code Page Maintenance Menu*. | all objects |
| **Password** | See **Password** in *Code Page Maintenance Menu*. | all objects |
| **Cipher** | See **Cipher** in *Code Page Maintenance Menu*. | all objects |
| **Profile types** | The type of error message to be processed:<br><br>S process short error messages only<br>L process long error messages only<br>*/A process all error messages | P |

### Assign Code Page Information to Objects

This function is used to assign a code page to an unassigned object. The code of this object is *not* converted to the specified code page.

You can also use the function to change the code page information for an object to which a code page is already assigned. In this case, only the code page name (IANA name) changes; the source code of this object is *not* converted.

The fields contained in the **Assign Code Page Information to Sources** screen are explained in the following table:

| Field | Explanation | Object |
|---|---|---|
| **Use selection list** | Specifies whether selective processing or automated processing is used for the specified objects: see *Object Selection List*. | all objects |
| **Forced assignment** | Specifies whether to process objects that have already code page information or objects without code page information.<br><br>Possible values are:<br><br>Y Yes.<br>Forced assignment is activated: the code page information changes to the specified code page for objects that have already code page information. | all objects |

| Field | Explanation | Object |
|---|---|---|
| | N No.<br>Forced assignment is deactivated (this is the default setting): the specified code page is only assigned to objects that have no code page information. | |
| **Code page** | The name of the code page to be assigned to the specified objects.<br><br>The default name entered is the IANA name as returned by the `*CODEPAGE` system variable (see the *System Variables* documentation). | all objects |
| **Object name/Object name range** | See **Object name/Object name range** in ***Check Conversion of Unassigned Objects***. | all objects |
| **Language code** | See **Language code** in ***Check Conversion of Unassigned Objects***. | E, S |
| **Short/long error msgs** | See **Short/long error msgs** in ***Check Conversion of Unassigned Objects***. | E, S |
| **Library** | See **Library** in ***Code Page Maintenance Menu***. | N, E |
| **DBID** | See **DBID** in ***Code Page Maintenance Menu***. | all objects |
| **FNR** | See **FNR** in ***Code Page Maintenance Menu***. | all objects |
| **Password** | See **Password** in ***Code Page Maintenance Menu***. | all objects |
| **Cipher** | See **Cipher** in ***Code Page Maintenance Menu***. | all objects |
| **Profile types** | See **Profile types** in ***Check Conversion of Unassigned Objects***. | P |

### Check Conversion of Assigned Objects

This function is used to test whether an assigned object can be converted from its current code page (as entered in the object directory information) to another code page.

An assigned object is an object which has code page information.

The fields contained in the **Check Conversion of Assigned** screen are explained in the following table:

| Field | Explanation | Object |
|---|---|---|
| **Use selection list** | Specifies whether selective processing or automated processing is used for the selected objects: see ***Object Selection List***. | all objects |
| **Current code page** | The code page or a range code pages to be used as an object selection criterion: see ***Name Specification*** for valid input values.<br><br>The default setting is asterisk (*) indicating all code pages. | all objects |
| **New code page** | The name of the code page to check for conversion of the specified assigned objects.<br><br>The default name entered is the IANA name as returned by the `*CODEPAGE` system variable (see the *System Variables* documentation). | all objects |

| Field | Explanation | Object |
|---|---|---|
| **Object name/Object name range** | See **Object name/Object name range** in *Check Conversion of Unassigned Objects*. | all objects |
| **Language code** | See **Language code** in *Check Conversion of Unassigned Objects*. | E, S |
| **Short/long error msgs** | See **Short/long error msgs** in *Check Conversion of Unassigned Objects*. | E, S |
| **Library** | See **Library** in *Code Page Maintenance Menu*. | N, E |
| **DBID** | See **DBID** in *Code Page Maintenance Menu*. | all objects |
| **FNR** | See **FNR** in *Code Page Maintenance Menu*. | all objects |
| **Password** | See **Password** in *Code Page Maintenance Menu*. | all objects |
| **Cipher** | See **Cipher** in *Code Page Maintenance Menu*. | all objects |
| **Profile types** | See **Profile types** in *Check Conversion of Unassigned Objects*. | P |

### Convert to Different Code Page

This function is used to convert an assigned object from its current code page (as entered in the object directory information) to another code page. You cannot convert an unassigned object.

The fields contained in the **Convert to Different Code Page** screen are explained in the following table:

| Field | Explanation | Object |
|---|---|---|
| **Use selection list** | Specifies whether selective processing or automated processing is used for the specified objects: see *Object Selection List*. | all objects |
| **Current code page** | The code page or a range code pages to be used as an object selection criterion: see *Name Specification* for valid input values.<br><br>The default setting is asterisk (*) indicating all code pages. | all objects |
| **New code page** | The name of the code page into which to convert the specified objects.<br><br>The default name entered is the IANA name as returned by the `*CODEPAGE` system variable (see the *System Variables* documentation). | all objects |
| **Object name/Object name range** | See **Object name/Object name range** in *Check Conversion of Unassigned Objects*. | all objects |
| **Language code** | See **Language code** in *Check Conversion of Unassigned Objects*. | E, S |
| **Short/long error msgs** | See **Short/long error msgs** in *Check Conversion of Unassigned Objects*. | E, S |
| **Library** | See **Library** in *Code Page Maintenance Menu*. | N, E |
| **DBID** | See **DBID** in *Code Page Maintenance Menu*. | all objects |
| **FNR** | See **FNR** in *Code Page Maintenance Menu*. | all objects |
| **Password** | See **Password** in *Code Page Maintenance Menu*. | all objects |

| Field | Explanation | Object |
|---|---|---|
| **Cipher** | See **Cipher** in *Code Page Maintenance Menu*. | all objects |
| **Profile types** | See **Profile types** in *Check Conversion of Unassigned Objects*. | P |

### Remove Code Page Information from Sources

This function is used to remove the code page information (as entered in the object directory) from an assigned object.

**Caution:** Be aware that the code page information is removed without conversion of the source code.

The fields contained in the **Remove Code Page Information from Sources** screen are explained in the following table:

| Field | Explanation | Object |
|---|---|---|
| **Use selection list** | Specifies whether selective processing or automated processing is used for the specified objects (see *Object Selection List*). | all objects |
| **Current code page** | The code page or a range code pages to be used as an object selection criterion: see *Name Specification* for valid input values. The default setting is the IANA name as returned by the *CODEPAGE system variable (see the *System Variables* documentation). | all objects |
| **Object name/Object name range** | See **Object name/Object name range** in *Check Conversion of Unassigned Objects*. | all objects |
| **Language code** | See **Language code** in *Check Conversion of Unassigned Objects*. | E, S |
| **Short/long error msgs** | See **Short/long error msgs** in *Check Conversion of Unassigned Objects*. | E, S |
| **Library** | See **Library** in *Code Page Maintenance Menu*. | N, E |
| **DBID** | See **DBID** in *Code Page Maintenance Menu*. | all objects |
| **FNR** | See **FNR** in *Code Page Maintenance Menu*. | all objects |
| **Password** | See **Password** in *Code Page Maintenance Menu*. | all objects |
| **Cipher** | See **Cipher** in *Code Page Maintenance Menu*. | all objects |
| **Profile types** | See **Profile types** in *Check Conversion of Unassigned Objects*. | P |

### Convert to Unshaped Form

This function is used to replace shaped Arabic characters (code page IBM420) with the corresponding unshaped characters in the following source objects: **Programming Objects** (N), **DDMs** (D), **User Error Messages** (E) and **System Error Messages** (S).

The fields contained in the **Convert Natural Sources to Unshaped Form** screen are explained in the following table:

| Field | Explanation | Object |
|---|---|---|
| **Use selection list** | Specifies whether selective processing or automated processing is used for the specified objects: see ***Object Selection List***. | N, D,E, S |
| **Current code page** | A code page or a range of code pages to be used as selection criterion for objects, see ***Name Specification*** for valid input values. The default is the IANA name as returned by the `*CODEPAGE` system variable, see the *System Variables* documentation. | N, D, E, S |
| **Object name** | See **Object name/Object name range** in *Check Conversion of Unassigned Objects*. | N, D, E, S |
| **Language code** | See **Language code** in *Check Conversion of Unassigned Objects*. | E, S |
| **Short/long error msgs** | See **Short/long error msgs** in *Check Conversion of Unassigned Objects*. | E, S |
| **Library** | See **Library** in *Code Page Maintenance Menu*. | N, E |
| **DBID** | See **DBID** in *Code Page Maintenance Menu*. | N, D, E, S |
| **FNR** | See **FNR** in *Code Page Maintenance Menu*. | N, D, E, S |
| **Password** | See **Password** in *Code Page Maintenance Menu*. | N, D, E, S |
| **Cipher** | See **Cipher** in *Code Page Maintenance Menu*. | N, D, E, S |

### Name Specification

You can specify a name or a range of names as a selection criterion.

In the list of options below, *value* is any combination of one or more characters:

| | Input | Items Selected |
|---|---|---|
| | *value* | All items with names equal to *value*. |
| | * | All items. |
| | ? | All items with any single character for each question mark (?) entered. |
| Leading characters | *value** | All items with names that start with *value*.<br><br>Example: `AB*`<br>Selected: AB, AB1, ABC, ABEZ<br>Not selected: AA1, ACB |
| Wildcard | *value*? | A wildcard. |

| | Input | Items Selected |
|---|---|---|
| | | All items with names that start with *value* and end with any single character for each question mark (?) entered.<br><br>Example: ABC?<br>Selected: ABCA, ABCZ<br>Not selected: AXC, ABCAA |
| | *value*? *value*?<br>*value***value*?<br>**value*?*value** | All items that match *value* combined with asterisk (*) and question mark (?) in any order.<br><br>Example: A?C*Z<br>Selected: ABCZ, AXCBBBZ, ANCZ<br>Not selected: ACBZ, ABDEZ, AXCBBBZA |
| Start value | *value*> | All items with names greater than or equal to *value*.<br><br>Example: AB><br>Selected: AB, AB1, BBB, ZZZZZZZ<br>Not selected: AA1, AAB |
| End value | *value*< | An end value: All items with names less than or equal to *value*.<br><br>Example: AX<<br>Selected: AB, AWW, AX<br>Not selected: AXA, AY |

### Object Selection List

You can set the **Use selection list** option to determine whether selective processing or automated processing is used for a maintenance function. If selective processing is used, a selection list of the specified objects is displayed on a selection screen before executing the function.

The **Use selection list** option does not apply to the **List Code Page Information of Sources** function.

Possible settings of **Use selection list** are as follows:

Y Yes.
Selective processing is activated (this is the default setting): a selection list of all objects that meet the specified selection criteria appears. You can then select the objects to be processed from this list.

N No.
Selective processing is deactivated and the function is executed immediately for all objects that meet the specified selection criteria.

An object selection list looks similar to the example shown below:

```
16:28:43                ***** NATURAL SYSCP UTILITY *****             2006-10-19
User SAG             - Check Conversion of Assigned Sources -
      Target code page IBM01140
Cmd  Name      Code Page                            Message
---  --------  -------------------------------- ------------------------------
__   LDA1      IBM01147
__   LDA2      IBM01147
__   LDA3      IBM01147
__   LDA4      IBM01147
__   MAP1      IBM01147
__   MAP2      IBM01147
__   MAP3      IBM01147
__   PGM2      IBM01147
__   PGM3      IBM01147
__   PROG1     IBM01147
__   PROG3     IBM01147
__   PROG4     IBM01147



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

The fields and columns contained in an object selection screen are described in the following table:

| Field/Column | Explanation | |
|---|---|---|
| **Target code page** | The code page to be used to check or perform a source-object assignment or conversion. | |
| **Cmd** | Input field for either of the following line commands to be executed for a selected object: | |
| | | |
| | `EX`<br>or<br>`X` | Execute the maintenance function.<br>You can press PF5 if you want to issue the line command to all objects in one go. |
| | | |
| | `LD` | Display object directory information.<br>This line command corresponds to the command `LIST DIRECTORY` *`object-name`* described in *Displaying Directory Information* in the *System Commands* documentation. |
| **Name** | The name of the object that meets the specified selection criteria. | |
| **Code Page** | The current code page information of the object. | |
| **Message** | This column only contains text when you have finished executing the maintenance function. In this case, the column contains a message that indicates the processing status of the object. See also ***Function Result Report***. | |

### Function Result Report

After a maintenance function has finished executing, the processing results are shown on a report screen. A report screen looks similar to an object selection screen an **example** of which is shown in *Object Selection List*.

The fields and columns contained in a result report screen are explained in the following table:

| Field/Column | Explanation |
|---|---|
| **Target code page** | The code page used to check or perform a source-object assignment or conversion. |
| **Cmd** | Input not possible. |
| **Name** | The name of the object that meets the specified selection criteria. |
| **Code Page** | The current code page information of the object. |
| **Message** | This column contains a message that indicates the processing status of the objects selected for processing. The messages indicate successful execution of a function or possible error reasons.<br><br>Possible messages are:<br><br>`Assignment possible` — The object can be assigned to the specified code page.<br>`Conversion error, at least one code point not translated.` — The object cannot be assigned or converted to the specified code page.<br>`Code page assigned` — The object has been assigned to the specified code page.<br>`Conversion possible` — The object can be converted to the specified code page.<br>`Code page converted` — The object has been converted from its current code page to another code page.<br>`Not converted` — The object has not been converted to the specified code page because it is already encoded in this code page. |

## All Code Pages

This function is used to list all code pages available in your current Natural environment as shown in the following example:

```
17:21:36                ***** NATURAL SYSCP UTILITY *****             2007-08-02
User SAG                       - All Code Pages -

Cmd Stat Name                                                          Units
--- ---- ------------------------------------------------------------- -----
 _   D   UTF-8                                                          1 - 3
 _   D   UTF-16                                                         2 - 2
 _   D   UTF-16BE                                                       2 - 2
 _   D   UTF-16LE                                                       2 - 2
 _   D   UTF-32                                                         4 - 4
 _   D   UTF-32BE                                                       4 - 4
 _   D   UTF-32LE                                                       4 - 4
 _   D   UTF16_PlatformEndian                                            2 - 2
 _   D   UTF16_OppositeEndian                                            2 - 2
 _   D   UTF32_PlatformEndian                                            4 - 4
 _   D   UTF32_OppositeEndian                                            4 - 4
 _   D   UTF-7                                                           1 - 4
 _   D   IMAP-mailbox-name                                               1 - 4
 _   D   SCSU                                                            1 - 3


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

You can use the following PF keys:

- PF8 (or ENTER) scrolls down one page in the list.
- PF7 scrolls up one page in the list.
- PF5 sorts the list in ascending order by code page name. Depending on the size of the list, you may have to increase the size of the sort buffer by using the `SORT` profile parameter as described in *SORT - Control of Sort Program* in the *Parameter Reference* documentation.

The columns contained in the **All Code Pages** screen are explained in the following table:

| Column | Explanation | |
|---|---|---|
| **Cmd** | Input field for one of the following line commands to be executed for the selected code page: | |
| | | |
| | `N` | Display all names used for the code page:<br><br>The IANA (Internet Assigned Numbers Authority) name is the standard and unambiguous name of the code page. The IANA name is used by Natural as the default code page name (see the `CP` profile parameter described in the *Parameter Reference* documentation) for conversions to and from Unicode. The IANA name is returned by the `*CODEPAGE` system variable (see the *System Variables* documentation). |

| Column | Explanation | |
|---|---|---|
| | | CCSID (Coded Character Set Identifier) denotes the character set as identified by IBM.<br><br>Alias names: one or more alternate names for the code page. |
| | | |
| | C | Display all code points of the selected code page: see ***Code Point List*** below. |
| | | |
| | T | Invoke a window to test code point conversion to and from Unicode: see ***Test Conversion*** below. |
| **Stat** | All code pages to be used during a Natural session must be predefined and enabled in the NATCONFG module.<br><br>This column shows the NATCONFG status of the code page: | |
| | | |
| | E | Code page is defined in the NATCONFG module and is enabled. |
| | | |
| | D | Code page is defined in NATCONFG but is disabled. |
| | | |
| | N | Code page is not defined in NATCONFG. |
| | | |
| | For detailed information on the NATCONFG module, refer to *Natural Configuration Tables* in the *Operations* documentation. | |
| **Name** | The internal ICU name. | |
| **Units** | The code units (minimum and maximum numbers of bytes) assigned to the code points. | |

This section covers the following topics:

- Code Point List
- Test Conversion

## Code Point List

This function is used to list all code points of the selected code page as shown in the following example:

```
13:38:33                ***** NATURAL SYSCP UTILITY *****             2007-08-06
+--------------------------- Code Points of UTF-8 ----------------------------+
! CP: 00000000  U: 0000        NULL                                           !
! CP: 00000001  U: 0001  ?     START OF HEADING                               !
! CP: 00000002  U: 0002  ?     START OF TEXT                                  !
! CP: 00000003  U: 0003  ?     END OF TEXT                                    !
! CP: 00000004  U: 0004  ?     END OF TRANSMISSION                            !
! CP: 00000005  U: 0005  ?     ENQUIRY                                        !
! CP: 00000006  U: 0006  ?     ACKNOWLEDGE                                    !
! CP: 00000007  U: 0007  ?     BELL                                           !
! CP: 00000008  U: 0008  ?     BACKSPACE                                      !
! CP: 00000009  U: 0009  ?     CHARACTER TABULATION                           !
! CP: 0000000A  U: 000A  ?     LINE FEED (LF)                                 !
! CP: 0000000B  U: 000B  ?     LINE TABULATION                                !
! CP: 0000000C  U: 000C  ?     FORM FEED (FF)                                 !
! CP: 0000000D  U: 000D  ?     CARRIAGE RETURN (CR)                           !
! CP: 0000000E  U: 000E  ?     SHIFT OUT                                      !
! CP: 0000000F  U: 000F  ?     SHIFT IN                                       !
+-----------------------------------------------------------------------------+
 _   D   ibm-912_P100-1995                                                1 - 1
 _   D   ibm-913_P100-2000                                                1 - 1
 _   N   ISCII,version=0                                                  1 - 4
 _   N   ISCII,version=1                                                  1 - 4


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

The list contains the following information:

- The byte sequence of the code page code points (`CP`).
- The byte sequence of the corresponding Unicode code points (`U`).
- The Unicode character. If the character cannot be interpreted by the current terminal emulation, the substitution character (as defined in the code page; here: `?`) is displayed instead.
- The normative name of the Unicode character.

The PF keys provided for each code point list are explained in the following table:

| PF Key | Function |
|---|---|
| PF4 | Not applicable to a code page with a 1-byte unit as the maximum.<br><br>Opens the **Leading Bytes of Code Point** window (see the relevant section) in which you can enter the byte range you want to view.<br><br>Press PF3 or ENTER to confirm your current input of leading bytes and to close the window.<br><br>Press PF12 to cancel your current input and to close the window. |

| PF Key | Function |
|---|---|
| PF5 | Invokes the **Unicode Properties** screen (see the relevant section) for the list item where the cursor is placed. |
| PF6 | Resets the first (non-leading) byte of the byte range to the hexadecimal value 0x00. |
| PF7 | Scrolls up one page in the selected byte range (see also *Specifying Leading Bytes*).<br><br>In a UTF-16 or UTF-32 code page, you can scroll through all byte ranges. |
| PF8 (or ENTER) | Scrolls down one page in the selected byte range (see also *Specifying Leading Bytes*).<br><br>In a UTF-16 or UTF-32 code page, you can scroll through all byte ranges. |
| PF10 | Moves to the leftmost screen position. |
| PF11 | Moves to the right of the screen. |

### Specifying Leading Bytes

This function does not apply to a code page with a 1-byte unit as the maximum.

You can use the **Leading Byte of Code Point** window to view the byte range (hexadecimal values 0x00 to 0xFF) of a particular leading byte for a code point.

In the following example of a UTF-8 code page, the hexadecimal values 0x22 and 0x32 have been entered as the leading bytes:

```
+--------- Leading Bytes of Code Point ---------+
!                                               !
!    Maximum number of bytes .. 3               !
!                                               !
!                                               !
!    Enter leading bytes ...... 00 22 32 00     !
!                                               !
!                                               !
!                                               !
+-----------------------------------------------+
```

After pressing PF3 (or ENTER) the code point list then displays the bytes from hexadecimal 0x00223200 to 0x002232FF.

**Note:** For byte-swapped code pages such as UTF-16LE or UTF-32LE, the bytes are read and displayed in a reversed byte order.

### Test Conversion

You can test code-point conversion from a selected code page to the default code page (value of `*CODEPAGE`) defined with the `CP` profile parameter:

- from an alphanumeric character string to Unicode code points and vice versa, or
- from hexadecimal values to Unicode code points and vice versa.

The example below shows the conversion window of a code page (here: `ibm-1140_P100-1997`) which contains the following information:

- the number of byte units (minimum and maximum numbers of bytes) assigned to the code points,
- an alphanumeric character string and its equivalent hexadecimal values and
- the corresponding Unicode code points.

```
+------------------- Test Conversion of ibm-1140_P100-1997 -------------------+
!                                                                              !
! Code page units .. 1 - 1   (minimum, maximum of bytes)                      !
!                                                                              !
!                                                                              !
! Code Page Characters                                                         !
! Alphanumeric .. ABC                                                          !
! Hexadecimal ... C1 C2 C3 40 40 40 40 40 40 40 40 40 40 40 40 40 40 40 40 40 !
!                 40 40 40 40 40 40 40 40 40 40 40 40 40 40 40 40 40 40 40 40 !
!                                                                              !
! Unicode                                                                      !
! Code points ... 0041 0042 0043 0020 0020 0020 0020 0020 0020 0020            !
!                 0020 0020 0020 0020 0020 0020 0020 0020 0020 0020            !
!                 0020 0020 0020 0020 0020 0020 0020 0020 0020 0020            !
!                 0020 0020 0020 0020 0020 0020 0020 0020 0020 0020            !
!                                                                              !
!                                                                              !
!                                                                              !
+------------------------------------------------------------------------------+
```

#### To convert a character or code point

1 Activate the field where you want to enter the literal string or code unit sequence to be converted:

   Press PF6 to enter a literal string in the **Alphanumeric** field (default input field).

   Or:

Press PF7 to enter hexadecimal values in the **Hexadecimal** field.

Or:

Press PF8 to enter Unicode code points in the **Unicode** field.

2 Press ENTER.

The value entered in one of the fields is converted to its equivalent code points or literal string.

## Unicode Properties

This function is used to display whether a Unicode character property is true (`yes`) or false (`no`) for a character contained in the default code page (value of `*CODEPAGE`) as shown in the example of the letter `A` in code page `IBM01140` below:

```
14:43:19               ***** NATURAL SYSCP UTILITY *****            2008-09-23
User SAG                    - Unicode Properties -

Default code page ... IBM01140

Alpha character ..... A     C1       hexadecimal    Substitution .. ? 3F
Unicode code point .. 0041

Unicode char. name .. LATIN CAPITAL LETTER A

Alphabetic .......... yes            Control ........ no
Alphanumeric ........ yes            Space .......... no
Lower case .......... no             Whitespace ..... no
Upper case .......... yes            Blank .......... no
Digit ............... no             Punctuation .... no
Hexadecimal ......... yes            Combining ...... no
Graphic ............ yes            Surrogate ...... no
Printable ........... yes            Right to left .. no


Command ===>


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

In the **Alpha character** field, you can enter the character whose properties you want to view. Press PF5 if you want to enter a Unicode code point.

For explanations of the Unicode character properties displayed on the screen, refer to Unicode Consortium's documentation *Unicode Character Database* at web site ***http://www.unicode.org/Public/4.1.0/ucd/UCD.html***.

## ICU Information

This function is used to display information on the ICU used:

| | |
|---|---|
| Product | Product name:<br><br>International Components for Unicode for Software AG (ICS) |
| Product code | ICS |
| Product version | ICS product version |
| Cumulated fix | Cumulated fix (if available and applied) for the current ICS product version |
| Architecture level | IBM architecture level if used on z/OS or z/VSE<br><br>The default is zero (`0`) denoting that no architecture level is used. |
| Revision | ICS revision number |
| ICU version | ICU version supported by ICS |
| Unicode version | Unicode version supported by ICS |
| Data library | Name of data library as assigned with `CFICU=(DATFILE=value)`. If more than one data library is active, a comma-delimited list is displayed containing the names of the active data libraries. |
| *CODEPAGE | Current code page used. Default is `IBM01140`.<br><br>See also system variable `*CODEPAGE`. |
| *LOCALE | Current locale used. Default is `en_US`.<br><br>See also system variable `*LOCALE`. |

**Note:** Similar information is displayed by the system command `CPINFO`.

# XIII SYSEDT Utility - Editor Buffer Pool Administration

# 71 SYSEDT Utility - Editor Buffer Pool Administration


- Defining a Natural Security Library Profile ..... 466
- Invoking SYSEDT and Executing a Function ..... 466
- General Information ..... 468
- Generation Parameters ..... 469
- Users ..... 470
- Logical Files ..... 471
- Recovery Files ..... 472
- Administration Facilities ..... 472
- Help on Direct Commands and Menu Functions ..... 473

The SYSEDT utility provides administration functions for the editor buffer pool, which is a data container for the Software AG editor. The SYSEDT utility can be used for the following:

- Displaying parameters and runtime information of the editor buffer pool.
- Modifying parameters that control and initialize the editor buffer pool and its work file.
- Deleting logical work files and recovery files.

The *SYSEDT Utility - Editor Buffer Pool Administration* documentation is associated with the following documentation:

- *Operations*: *Operating the Software AG Editor*
- *Parameter Reference*: *EDBP - Software AG Editor Buffer Pool Definitions*

## Defining a Natural Security Library Profile

If you have Natural Security installed, you must create a library security profile for the SYSEDT utility.

For details, see *Library Maintenance* described in the *Natural Security* documentation.

## Invoking SYSEDT and Executing a Function

This section provides instructions for invoking the SYSEDT utility and executing a SYSEDT utility function.

### To invoke the SYSEDT utility

- Issue the following Natural system command:

  ```
  SYSEDT
  ```

  A SYSEDT **Main Menu** similar to the example below appears:

```
 15:47:50              ***** NATURAL SYSEDT UTILITY *****             2009-01-29
 User MMO                        - Main Menu -                    TID 1     33



                    Code   Function

                      G    General Information
                      P    Generation Parameters
                      U    Users
                      F    Logical Files
                      R    Recovery Files
                      A    Administration Facilities
                      .    Exit

             Code ..




Command ===>
Enter-PF1---PF2---PF3---PF4---PF5---PF6---PF7---PF8---PF9---PF10--PF11--PF12---
Cont  Help  Menu  Exit        Admin Files Users Recov       GInfo Parms Canc
```

The functions available in the menu are explained in the following sections.

### To execute a SYSEDT function

- In the **Code** field of the SYSEDT **Main Menu**, enter the one-character code that corresponds to the function you want to execute and press ENTER. For example:

  ```
  G
  ```

  Executes the **General Information** function.

  Or:

  In the SYSEDT **Main Menu**, press the PF key that corresponds to the function you want to execute. For example:

  PF10 (GInfo)

  Executes the **General Information** function.

  Or:

  Issue the `SYSEDT` system command followed by any function code available in the SYSEDT **Main Menu**. For example:

  ```
  SYSEDT G
  ```

Executes the **General Information** function.

For a list of all SYSERR direct commands available, use the help function described in ***Help on Direct Commands and Menu Functions***.

## General Information

This function invokes the **General Information** screen which provides an overview of the current status of the editor buffer pool. The fields contained on the screen are described in the following table. For detailed information on these fields, see *Editor Work File* and *Editor Buffer Pool* in the *Operations* documentation.

| Field | Explanation |
|---|---|
| **Usage Statistics** | The currently available total number, the currently used number, and the currently used percentage of the available number of the items that follow. |
| **Buffer Pool Blocks** | The number of blocks in the editor buffer pool. See also *Obtaining Free Blocks*.* |
| **Work File Records** | The number of records in the editor work file. See also *Editor Work File*.* |
| **Control** | The number of control records, which is always one. See also *Control Record*.* |
| **Work** | The number of work records. See also *Work Record* .* |
| **Recovery** | The number of recovery records. See also *Recovery Records*.* |
| **Logical Files** | The number of logical files. |
| **Requests** | The total number of read and write requests, the number of read and write requests for buffer pool blocks (**Pool** column), and the number of read and write requests for work or recovery files (**File** column). The **Copy** column shows read requests, which (in contrast to locked read requests) result in the deletion of the corresponding buffer pool block. |
| **Read Work** | The number of read requests for logical file records. A logical file record can be found in the buffer pool (**Pool** column) or on the work file (**File** column). It can be read by a locked or by a copy request: locked means that the record is kept in the buffer pool for some time; copy means that it is deleted from the buffer pool after having been read. |
| **Write Work** | The number of write requests for logical file records. A record can be either written to the buffer pool (**Pool**) or moved to the work file (**File**) if there are no free blocks available. |
| **Read Recovery** | The number of read requests for recovery records in the editor work file. |
| **Write Recovery** | The number of write requests for recovery records in the editor work file. |
| **Timeout Values in Seconds** | Items with timeout values specified in seconds. These timeout values can be modified after pressing PF5 (Updat) and dynamically set after pressing PF5 (Save) again. The modified values are not kept during a restart of the buffer pool. The values from the work file control record are used instead. See also *Obtaining Free Blocks*.* |

| Field | Explanation |
|---|---|
| **Logical Files** | The time after which a logical file is deleted if it has not been accessed during this time. |
| **Files Delete Check** | The time after which all logical files are checked periodically whether they can be deleted. |
| **Changed Blocks** | The time after which blocks that have been modified can be freed by writing them to the work file. |
| **Unchanged Blocks** | The time after which blocks that have not been modified can be freed by writing them to the work file. |
| **Locked Blocks** | The time after which blocks that have been read with locked can be freed by writing them to the work file. |

* described in the *Operations* documentation

## Generation Parameters

This function invokes the **Generation Parameters** screen where you can view and change the current parameter settings of the editor buffer pool:

| Parameter | Explanation |
|---|---|
| DDNAME | The name of the editor work file for the JCL definition. |
| DSNAME | The name of the work file data set (file).<br><br>This parameter is not displayed in a Com-plete or z/VSE batch environment. |
| RECNUM | The total number of work file records. |
| LRECL | The work file record length. |
| PWORK | The percentage of work file records used as work records. |
| RWORK | The percentage of work records for regular logical files. |
| MAXLF | The maximum number of logical files in the editor buffer pool. |
| FTOUT | The timeout value (in seconds) for a logical file, which has not been accessed, to be deleted. |
| DTOUT | The period of time (in seconds) for logical files to be checked for deletion. |
| CTOUT | The timeout value (in seconds) for changed buffer pool blocks. |
| UTOUT | The timeout value (in seconds) for unchanged buffer pool blocks. |
| LTOUT | The timeout value (seconds) for locked buffer pool blocks. |
| ITOUT | The timeout value (in seconds) for the buffer pool initialization. |

The parameters above are keyword subparameters of the EDBP profile parameter (or corresponding NTEDBP macro), which is used to define the initial editor buffer pool settings. For detailed information on EDBP and all keyword subparameters available, see *EDBP - Software AG Editor Buffer Pool Definitions* in the *Parameter Reference* documentation.

The **Start** column on the **Generation Parameters** screen refers to the buffer pool restart (see also *Initializing the Editor Buffer Pool* in the *Operations* documentation). The following start values can be displayed for the parameters:

| Value | Explanation |
|---|---|
| L | The value for the corresponding parameter is taken from either the editor parameter module or the work file definition. |
| C | A modification of the corresponding parameter value forces a buffer pool cold start. Recovery records are lost. |
| W | A modification of the corresponding parameter value results in a buffer pool warm start. Recovery records are kept. |

### To modify parameter values

1 On the **Generation Parameters** screen, press PF5 (Updat).

The write-protected fields next to the parameters change to input fields.

2 Replace the parameter value(s) required and press PF5 (Save) to save the new values in the control record of the editor work file. They will be activated when the editor buffer pool is started again.

If you change the value of PWORK, a window prompts you to confirm the cold start of the editor work file. Enter YES to confirm the new parameter value and to execute a buffer pool cold start for the work file initialization. (Any other entry or no entry resets the parameter to its previous value.)

## Users

This function invokes the **Users** screen which provides the following information:

| Column | Explanation |
|---|---|
| **User ID** | The Natural user ID. |
| **Logical Files** | The number of logical files defined per user. |
| **Pool Blocks** | The number of buffer pool blocks per user. |
| **Work Records** | The number of work records per user. |
| **Recovery Files** | The number of recovery files per user. |
| **Recovery Records** | The number of recovery records per user. |

For each user listed, you can execute one of the following line commands. You enter a line command in the **C** column, next to the user required.

| Line Command | Explanation |
|---|---|
| ?<br><br>or<br>* | Opens a window with a list of all valid line commands and corresponding PF keys, if available. |
| . | Exits the current screen and returns to the previous screen. |
| /<br><br>or<br>P | Positions the line to the top of the screen. |
| F | Selects the logical files of this user. |
| R | Selects the recovery files of this user. |
| D | Deletes all logical files and/or recovery files for this user. |

## Logical Files

This function invokes the **Logical Files** screen which provides the following information:

| Column | Explanation |
|---|---|
| **File No.** | The logical file number. |
| **User ID** | The Natural user ID. |
| **Type** | The logical file type. |
| **Pool Blks** | The number of buffer pool blocks currently used per logical file. |
| **File Recs** | The number of work file records currently allocated per logical file. |
| **Last Access** | The date and time of the last read or write request per logical file. |

For each logical file listed, you can execute one of the following line commands. You enter a line command in the **C** column, next to the user required.

| Line Command | Explanation |
|---|---|
| ?<br><br>or<br>* | Opens a window with a list of all valid line commands and corresponding PF keys, if available. |
| . | Exits the current screen and returns to the previous screen. |
| /<br><br>or<br>P | Positions the line to the top of the screen. |
| S | Selects the logical files of this user. |
| D | Deletes the logical file. |

## Recovery Files

This function invokes the **Recovery Files** screen which provides the following information:

| Column | Explanation |
|---|---|
| **User ID** | The Natural user ID. |
| **Member** | The library member name. |
| **Library** | The library name. |
| **Type** | The library type. |
| **Recs** | The number of recovery records per recovery file. |
| **Creation Date** | The creation date of the recovery file. |
| **Creation Time** | The creation time of the recovery file. |

For each recovery file listed, you can execute one of the following line commands. You enter a line command in the **C** column, next to the user required.

| Line Command | Explanation |
|---|---|
| ?<br>or<br>* | Opens a window with a list of all valid line commands and corresponding PF keys, if available. |
| . | Exits the current screen and returns to the previous screen. |
| /<br>or<br>P | Positions the line to the top of the screen. |
| S | Selects the recovery files of this user. |
| D | Deletes the recovery file. |

## Administration Facilities

This function invokes the **Administration Facilities** screen which can be used to terminate the editor buffer pool or the SYSEDT utility, or display the editor buffer pool in dump (hexadecimal) format for problem diagnosis.

### To use the Administration Facilities screen

- Terminate the editor buffer pool, by entering function code T.

A window prompts you for confirmation:

- If you enter `YES`, an additional window opens, asking you whether or not you want the editor buffer pool to be immediately restarted.

  If you enter `YES` again, the buffer pool is immediately restarted, which gives you the possibility to immediately activate modified generation parameters.

  If you enter `NO`, you leave SYSEDT and can perform actions outside your TP environment, for example, change the size of your editor work file.

Or:

Display the storage dump of the editor buffer pool, by entering function code `D`.

The offsets and addresses of the editor buffer pool are displayed on the screen.

If required, press `PF10` (RelO) to position offset `00000000` to the top of the screen. `PF1` (Help) invokes a list of all PF keys available.

Or:

Return to the SYSEDT **Main Menu** by entering a period (.).

## Help on Direct Commands and Menu Functions

You can obtain online help information on all SYSEDT direct commands available or on the function currently used.

### To view all SYSEDT direct commands

- In the command line of any SYSEDT utility screen, enter one of the following characters:

  ```
  ?
  ```

  or

  ```
  *
  ```

  All direct commands available within the SYSEDT utility are listed in alphabetical order.

### To view help on a specific SYSEDT function

- In the command line of a SYSEDT utility screen invoked by a SYSEDT utility function, enter the following:

  ```
  H
  ```

Or:

Press PF1 (Help).

Help information on the fields and columns contained on the current SYSEDT utility screen is displayed.

# XIV SYSERR Utility

When you develop a Natural application, you may want to separate error or information messages from your Natural code and manage them separately. This makes it easy for you, for example, to standardize messages, to have predefined message ranges for different types of message, to translate messages into other languages or to attach to a message a long text that explains it in more detail.

The SYSERR utility provides the option to write application-specific messages. In addition, you can use the SYSERR utility to customize the texts of the existing Natural system messages.


**General Information on Messages**

**Invoking SYSERR**

**Functions**

**Parameters**

**Direct Commands**

**Upper Case Conversion- ERRUPPER**

**Replacing Characters - ERRCHAR**

**Managing Messages in Different Libraries**

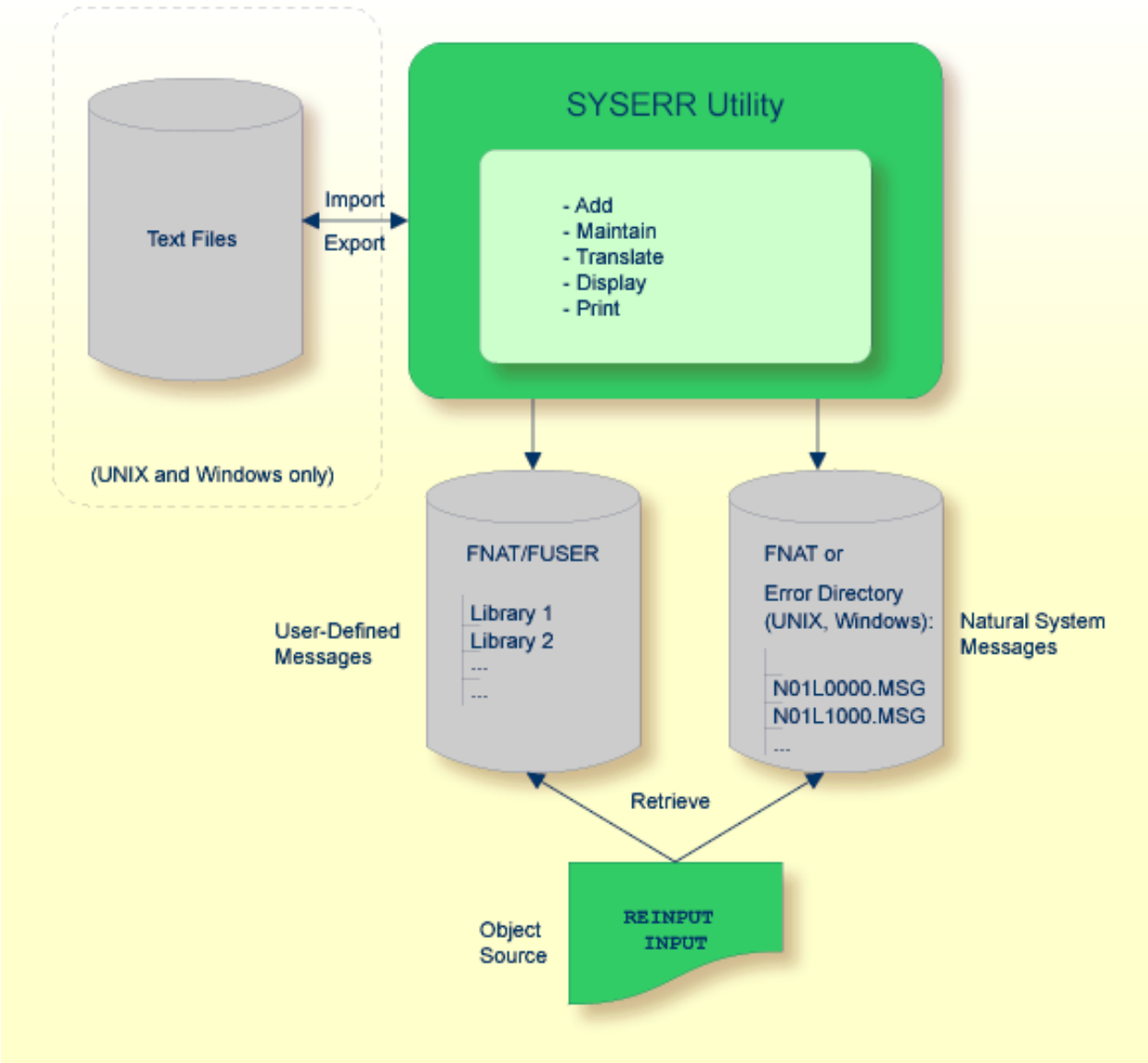
**Application Programming Interface USR0020P**

# 72 General Information on Messages


- Message Types ..... 479
- Message Languages ..... 479
- Messages and Code Page Support ..... 480
- Issuing Messages ..... 480
- Retrieving Natural System Short Messages ..... 481
- Retrieving User-Defined Short Messages ..... 481
- Obtaining Message Information ..... 482

This section contains information on the types of message and message languages that can be managed with the SYSERR utility and how messages are issued and retrieved in your Natural system environment.

The following graphic illustrates the features of the SYSERR utility and how messages are processed within Natural:



## Message Types

There are two types of message: Natural (system) messages and user-defined messages:

Natural system messages are issued by the Natural nucleus and Natural utilities. Natural system messages are delivered by Software AG and stored as message files in the Natural system file FNAT. Natural system messages begin with `NAT`, followed by a four-digit number, for example, NAT0230.

User-defined messages are issued by applications written by a user. User-defined messages are stored as message files in libraries (including SYS-libraries) in the system file FUSER or FNAT.

A message can be translated into different languages. Each language is stored in a separate message file. A maximum of 9999 messages can be stored per library and message file.

There are four types of message text:

- Natural system short message
- Natural system long message
- User-defined short message
- User-defined long message

A short message is the one-line message which is displayed in the message line when the corresponding error situation occurs.

A long message is a detailed explanation of the corresponding short message and includes instructions for solving problems.

**Caution:** Keep in mind that any modifications of Natural system messages can result in wrong messages or a loss of modifications when a new Natural version is released.

## Message Languages

Messages can be created in up to 60 languages as described for the system variable `*LANGUAGE` in the *System Variables* documentation.

The following rules and restrictions apply:

- Natural system short messages must be entered in English first, and can then be translated into any other language.
- Natural system long messages can be entered in English, but cannot be translated into other languages.

- User-defined short messages can be entered in any language, and then be translated into any other language.
- User-defined long messages can be entered in any language, but only if the corresponding short message in the same language already exists.

## Messages and Code Page Support

For every short and long message a code page may be saved.

With session parameter `SRETAIN=OFF` the current session code page (*CODEPAGE) is assigned to an error message when it is saved.

When reading an error message with SYSERR with session parameter `SRETAIN=OFF`, an error message will be converted into the current session code page. For further information see Code Page Support for Editors, System Commands and Utilities in section *Development Environment* of the *Unicode and Code Page Support* documentation.

## Issuing Messages

This section contains information on the Natural statements `INPUT` and `REINPUT` that are used to issue a Natural system short message or a user-defined short message in a Natural program.

### To issue a Natural system short message in a program

- Specify one of the following Natural statements:

  ```
  INPUT WITH TEXT *-nnnn ' '
  ```

  or

  ```
  REINPUT WITH TEXT *-nnnn
  ```

  where *nnnn* is the number of the requested message (you can omit leading zeros).

### To issue a user-defined short message in a program

- Specify one of the following Natural statements:

```
INPUT WITH TEXT *nnnn ' '
```

or

```
REINPUT WITH TEXT *nnnn
```

where *nnnn* is the number of the requested message (you can omit leading zeros).

### Dynamic Replacement of Message Text

A message text can contain variable parts that are identified by the notation `:n:`, where *n* represents occurrences 1 to 7. These variable parts are replaced by a value at runtime.

For details, see *operand3* in the section *INPUT Syntax 1 - Dynamic Screen Layout Specification* and *operand3* in the section *REINPUT* in the *Statements* documentation.

## Retrieving Natural System Short Messages

When a program references a Natural system short message, Natural looks for the requested message number in the Natural system file FNAT in the following order:

1. Under the current language code as determined by the system variable `*LANGUAGE`,
2. Under language code 1 (English).

If neither of the above is found, a program references a message that does not exist and you only receive the message number prefixed with `NAT`, for example, NAT0230.

## Retrieving User-Defined Short Messages

When a program references a user-defined short message, Natural first looks for the requested message number *nnnn* under the current language code as determined by the system variable `*LANGUAGE` (see the *System Variables* documentation). If that message does not exist, Natural looks for the requested message number *nnnn* under language code 1 (English). If that message does not exist either, Natural looks for message number *n*000 (where *n* is the first digit of the requested message number) under language code 1.

These three search steps are first performed in the current library. If nothing is found there, further libraries are searched in the same way until a corresponding message is found.

The sequence of libraries for the search is as follows:

1. The current library as determined by the system variable `*LIBRARY-ID`,

2. The steplibs; if Natural Security is installed, the sequence in which the steplibs are specified in the Natural Security profile of the current library,
3. The default steplib as determined by the system variable `*STEPLIB`,
4. The library SYSTEM in the system file FUSER (*),
5. The library SYSTEM in the system file FNAT (*).

(*) If the name of the current library begins with SYS, SYSTEM FNAT is searched before SYSTEM FUSER.

## Obtaining Message Information

When you receive a short message, you may be looking for additional information on the problem situation.

- With the system command `HELP`, you can display Natural system long messages or user-defined long messages.
- With the system command `LASTMSG`, you can list the short test of the message(s) that occurred last and additional information on the error situation. The information displayed includes associated error messages that possibly preceded the last message.

Both commands are described in the *System Commands* documentation.

# 73 Invoking SYSERR

### To invoke the SYSERR utility

- Enter the following system command:

  ```
  SYSERR
  ```

  The main menu of the SYSERR utility is displayed:

```
16:15:35              ***** NATURAL SYSERR UTILITY *****             2008-09-18
                                    - Menu -

                 Code  Function
                 ----  ----------------------------------------
                  AD   Add new messages
                  DE   Delete messages
                  DI   Display messages
                  MO   Modify messages
                  PR   Print messages
                  SC   Scan in messages
                  SE   Select messages from a list
                  TR   Translate messages into another language
                  ?    Help
                  .    Exit
                 ----  ----------------------------------------
          Code .. __   Message type .... US
                       Library ......... SYSTEM__
                       Message number .. 1___ - 9999
                       Language codes .. 1________

Command ===>









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

From the main menu of the SYSERR utility, you can execute all SYSERR utility functions available for creating and maintaining messages. The individual functions are explained in the section ***Functions***. The parameters that apply with the functions are explained in general in the section ***Parameters***; any restrictions that apply to the use of these parameters are described for each function concerned in the section *Functions.*

The SYSERR utility provides an extensive online help system. To obtain field-specific help information, either enter a question mark in the relevant field and press ENTER or place the cursor in the field and press PF1.

# 74 Functions


- Adding Messages ..... 486
- Deleting Messages ..... 490
- Displaying Messages ..... 490
- Modifying Messages ..... 492
- Printing Messages ..... 494
- Scanning Messages ..... 496
- Selecting Messages from a List ..... 498
- Translating Messages into other Languages ..... 501

You invoke a SYSERR utility function by entering the code that corresponds to the required function and one or more parameters in the input fields of the SYSERR main menu. This section describes the functions provided in the menu and the parameters that can be specified for each function. For general instructions on the use of parameters, see the section ***Parameters***.

## Adding Messages

### To add new messages

1 The SYSERR utility is case-sensitive by default. If you want lower to upper case translation for the messages to be created, enter the following terminal command:

```
%U
```

Any lower case characters you type when adding message text are then converted to upper case characters for the duration of the current Natural session. Message text is also converted to upper case if the profile parameter `TS` is set to `ON` (see the *Parameter Reference* documentation).

For detailed information on `%U`, see the *Terminal Commands* documentation.

2 Invoke the SYSERR main menu and enter the following values:

| Field | Input Value |
|---|---|
| **Code** | `AD` |
| **Message type** | `NS` Natural system short messages<br>`NL` Natural system long messages<br>`US` User-defined short messages<br>`UL` User-defined long messages<br><br>A long message can only be added if the corresponding short message already exists, as the long message is intended to be an explanation of the short message. |
| **Library** | Any existing Natural library. |
| **Message number** | Two numbers of up to four digits corresponding to the first and last numbers of the range of messages to be added. If you only want to add one message, either enter the number of the new message in the left **Message number** field and clear the right field, or enter the number in both fields. |
| **Language codes** | The code of the language for which the message is to be added. If the message type is `NS` or `NL`, the language code must be `1` for English. For other message types, the first language code entered in the field is used; all others are ignored. |

3 Press ENTER.

An **Add Short Message** screen similar to the example below is displayed:

```
15:56:01              ***** NATURAL SYSERR UTILITY *****            2008-11-28
Lower Case ON                - Add Short Message -

Number        Short Message (English)
------------  ---------------------------------------------------------------
SYSERR1004
              ....+....1....+....2....+....3....+....4....+....5....+..

Sample ...... Message sample number 0000 ..... ..... ..... ..... ..... .....
.....

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

The **Number** field indicates the message number (in the example above, `1004`), which is prefixed with the library ID (in the example above, `SYSERR`).

4 In the input line next to the message number, type in a short message text and press ENTER.

Or:

If the line labeled **Sample** contains a sample message text as shown in the example above, copy this text into the input line by entering `.C` and then pressing ENTER. If the sample message text contains the string `0000`, this string is replaced by the new message number as illustrated in the following example:

```
15:30:46              ***** NATURAL SYSERR UTILITY *****            2008-11-28
Lower Case ON                - Add Short Message -

Number        Short Message (English)
------------  ---------------------------------------------------------------
SYSERR1004    Message sample number 1004
              ....+....1....+....2....+....3....+....4....+....5....+..

Sample ...... Message sample number 0000









Message has been added.









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

For instructions on creating a sample message, see the `SAMPLE` command described in *Direct Commands*.

5 Press PF9 to add a corresponding long message text.

An **Add Long** *`message-number`* screen similar to the example below appears:

```
16:20:24       - Add Long SYSERR1004 (English) - Lower Case ON       2008-11-28
 1 Tx. Message sample number 1004
 2     .
 3     .
 4 Ex. .
 5     .
 6     .
 7     .
 8     .
 9     .
10     .
11     .
12     .
13     .
14     .
15     .
16     .
17     .
18 Ac. .
19     .
20     .









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

6 Enter text in the three input areas: **Tx.** (text), **Ex.** (explanation) and **Ac.** (action).

7 Press ENTER to save the long message.

8 Press PF9 to return to the short message or to add the next short message in ascending order if you selected a range of message numbers.

9 Press PF3 or PF12 to return to the SYSERR main menu.

Or:

Press PF8 or PF7 to add the next short message in ascending or descending order if you selected a range of message numbers.

## Deleting Messages

### To delete messages

- In the fields of the SYSERR main menu, enter the following values:

| Field | Input Value |
|---|---|
| **Code** | DE |
| **Message type** | NS Natural system short messages<br>NL Natural system long messages<br>US User-defined short messages<br>UL User-defined long messages<br><br>It is possible to delete a long message without deleting the corresponding short message, but not vice versa. If you try to delete a short message for which a long message exists, you are asked to confirm the deletion of both. |
| **Library** | Any existing Natural library. |
| **Message number** | Two numbers of up to four digits corresponding to the first and last numbers of the range of messages to be deleted. |
| **Language codes** | The code(s) of the language(s) in which the messages are to be deleted. To indicate that the messages specified are to be deleted in all languages available, enter an asterisk (*). |

## Displaying Messages

### To display messages

1 In the fields of the SYSERR main menu, enter the following values:

| Field | Input Value |
|---|---|
| **Code** | DI |
| **Message type** | NS Natural system short messages<br>NL Natural system long messages<br>US User-defined short messages<br>UL User-defined long messages |

| Field | Input Value |
|---|---|
| **Library** | Any existing Natural library. |
| **Message number** | Two numbers of up to four digits corresponding to the first and last numbers of the range of messages to be displayed. |
| **Language codes** | The code of the language in which the messages are to be displayed. Only one language code is accepted. If more than one code is specified, only the first one is used; all others are ignored. |

2 Press ENTER.

For short messages, a **Display Short Messages** screen similar to the example below appears:

```
15:41:11              ***** NATURAL SYSERR UTILITY *****            2008-11-28
                          - Display Short Messages -

Number        Short Message (English)
------------  ---------------------------------------------------------------
NAT0001       Missing/invalid syntax; undefined variable name/keyword.
NAT0002       No file is available with specified name or number.
NAT0003       Invalid character string for file name or file number.
NAT0004       DEFINE DATA must be the first statement if present.
NAT0005       Closing parenthesis missing in arithm/logical expression.
NAT0006       ESCAPE statement used when no processing loop active.
NAT0007       Invalid THRU or TO clause in READ LOGICAL or HISTOGRAM.



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

Press PF8 to page forwards.

For long messages, the **Display Long Message** screen is displayed where the messages are displayed one after another by pressing PF8 to page forwards or PF7 to page backwards. The **Display Long Message** screen is similar to the **Modify Long Message** screen shown in *Modifying Messages*.

# Modifying Messages

### To modify messages

1 The SYSERR utility is case-sensitive by default. If you want lower to upper case translation for the messages to be modified, enter the following terminal command:

```
%U
```

Any lower case characters you type when editing message text are then converted to upper case characters for the duration of the current Natural session. Message text is also converted to upper case if the profile parameter `TS` is set to `ON` (see the *Parameter Reference* documentation).

For detailed information on `%U`, see the *Terminal Commands* documentation.

2 In the fields of the SYSERR main menu, enter the following values:

| Field | Input Value |
|---|---|
| **Code** | `MO` |
| **Message type** | `NS` Natural system short messages<br>`NL` Natural system long messages<br>`US` User-defined short messages<br>`UL` User-defined long messages |
| **Library** | Any existing Natural library. |
| **Message number** | Two numbers of up to four digits corresponding to the first and last numbers of the range of messages to be modified. |
| **Language codes** | The code of the language in which the messages are to be modified. Only one language code is accepted. If more than one code is specified, only the first one is used; all others are ignored. |

3 Press ENTER.

A **Modify Short Message** screen similar to the example below is displayed:

```
16:46:31              ***** NATURAL SYSERR UTILITY *****            2008-11-28
Lower Case ON             - Modify Short Message -


Number        Short Message (English)
------------  ---------------------------------------------------------------
SYSERR1004    Message sample number 1004
              ....+....1....+....2....+....3....+....4....+....5....+..


 1 Tx. Input missing.
 2     .
 3     .
 4 Ex. Input value missing in field XYZ.
 5     Enter an alphanumeric value.
 6     .
 7     .
 8     .
18 Ac. Enter value in field XYZ.
19     .
20     .

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

For reference purposes, the long message is displayed in the bottom half of the screen.

When you modify long messages, the **Modify Long** *message-number* screen is displayed:

```
16:56:08     - Modify Long SYSERR1004 (English) - Lower Case ON      2008-11-28
 1 Tx. Input missing.
 2     .
 3     .
 4 Ex. Input value missing in field XYZ.
 5     Enter an alphanumeric value.
 6     .
 7     .
 8     .
 9     .
10     .
11     .
12     .
13     .
14     .
15     .
16     .
17     .
18 Ac. Enter value in field XYZ.
19     .
20     .

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

4 Press ENTER to save any modifications.

5 Press PF8 or PF7 to modify the next message in ascending or descending order if you selected a range of numbers.

## Printing Messages

### To print messages

1 In the fields of the SYSERR main menu, enter the following values:

| Field | Input Value |
|---|---|
| **Code** | PR |
| **Message type** | NS Natural system short messages<br>NL Natural system long messages<br>US User-defined short messages<br>UL User-defined long messages |
| **Library** | Any existing Natural library. |

| Field | Input Value |
|---|---|
| **Message number** | Two numbers of up to four digits corresponding to the first and last numbers of the range of messages to be printed. |
| **Language codes** | The code of the language in which the messages are to be printed. Only one language code is accepted. If more than one code is specified, only the first one is used; all others are ignored. |

2 Press ENTER.

A print window similar to the example below opens:

```
+------Print Natural System Messages-------+
!                                          !
!   Language code .... 1                   !
!                                          !
!   Long texts, too .. N                   !
!   Message number ... 1___ -  25          !
!   Lines per page ... 60_                 !
!   Left margin ...... 10                  !
!   Top margin ....... 0_                  !
!   Bottom margin .... 0_                  !
!   Printer ID ....... PRT1____            !
!                                          !
!                                          !
+------------------------------------------+
```

3 Specify the options provided in the print window and the logical printer name.

See the `DEFINE PRINTER` statement in the *Natural Statements* documentation for details on logical printer names.

4 Press ENTER to output the selected messages on a printer.

### To print all Natural system messages

- In the fields of the SYSERR main menu, enter the following values:
  - Code `PR`,
  - Message type `NS` or `NL`,
  - Message number range `1 - 9999`,
  - Language code `1` (English) or `2` (German).

  A library ID is not required and possible entries are ignored.

# Scanning Messages

This function is used to scan messages for a specific string of characters. Only short messages can be scanned.

### To scan messages

1 In the fields of the SYSERR main menu, enter the following values:

| Field | Input Value |
|---|---|
| **Code** | SC |
| **Message type** | NS Natural system short messages<br>US User-defined short messages |
| **Library** | Any existing Natural library. |
| **Message number** | Two numbers of up to four digits corresponding to the first and last numbers of the range of messages to be scanned. |
| **Language codes** | Specify a maximum of nine language codes from the ranges 1 - 9, A - Z and a - y, or enter an asterisk (*) for all languages. |

2 Press ENTER.

A scan window similar to the example below opens:

```
+-------------------------------------+
|   Scan value(s)       Or/And/Not    |
|   ----------------    ----------    |
|   ________________       OR_        |
|   ________________                  |
|   ________________                  |
|   ________________                  |
|                                     |
|   Absolute ..... X                  |
|   Immediate .... _                  |
|                                     |
+-------------------------------------+
```

In the fields provided, you can specify the search criteria to be used for scanning:

| | |
|---|---|
| **Scan value(s)** | In the four empty fields, enter up to four character strings to be searched for. The scan finds the specified terms in both upper and lower case. |
| **Or/And/Not** | You can perform a Boolean search query by entering one of the following operators:<br><br>`OR` — Searches for one or more of the character strings entered in **Scan value(s)**. This is the default setting.<br>`AND` — Searches for all of the character strings entered in **Scan value(s)**.<br>`NOT` — Searches for none of the character strings entered in **Scan value(s)**.<br><br>The operator is ignored if you only fill one of the **Scan value(s)** fields. |
| **Absolute** | If you mark this field, the string of characters is found even if it is part of a word. For example, if you scan for the value `meter`, the search would also find words such as parameter and millimeter.<br><br>If you remove the mark, the search is restricted to match entire words only. |
| **Immediate** | If you mark this field, messages are displayed individually, one after another. Otherwise, a list of messages is displayed after the search is completed.<br><br>If you specify more than one language or an asterisk (*) in the **Language codes** field, **Immediate** must be marked. |

3 Specify search criteria as shown in the following example:

```
+-------------------------------------+
!   Scan value(s)       Or/And/Not     !
!   ----------------    ----------     !
!   BUFFER___________       AND        !
!   POOL_____________                  !
!   _________________                  !
!   _________________                  !
!                                      !
!   Absolute ..... X                   !
!   Immediate .... _                   !
!                                      !
+-------------------------------------+
```

In the example above, the scan finds all short messages that contain both the words `buffer` and `pool`.

4 Press ENTER.

All messages to which the specified search criteria apply are listed on the screen as shown in the following example:

```
11:32:27              ***** NATURAL SYSERR UTILITY *****            2008-11-28
                          - Scan in Short Messages -

Number        Short Message (English)
------------- ------------------------------------------------------------------
NAT0777       Buffer pool full.
```

```
End of scan reached.

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

The word in which the search string is found is highlighted.

From this screen, you can display the search criteria used for the current scan by pressing PF10.

## Selecting Messages from a List

This function is used to display a range of messages and select single ones for further processing. Only short messages can be displayed.

### To select messages

1 In the fields of the SYSERR main menu, enter the following values:

| Field | Input Value |
|---|---|
| **Code** | SE |
| **Message type** | NS Natural system short messages<br>US User-defined short messages |
| **Library** | Any existing Natural library.<br><br>If an asterisk (*) is appended to the library ID, a list of all libraries available is displayed for selection. |
| **Message number** | Two numbers of up to four digits corresponding to the first and last numbers of the range of messages to be displayed for selection. |
| **Language codes** | The code of the language in which the messages are to be displayed.<br><br>If more than one code is specified, only the short message text of the first one is displayed. Enter an asterisk (*) to display the languages available for each message. |

2 Press ENTER.

A **Select Messages** screen similar to the example below is displayed:

```
17:08:13             ***** NATURAL SYSERR UTILITY *****              2008-11-28
                              - Select Messages -
                                                                  Languages
Se Number       Short Message (English)                           short     long
-- ------------ ------------------------------------------------ --------- ----
__ NAT0001      Missing/invalid syntax; undefined variable name/ 1         1
__ NAT0002      No file is available with specified name or numb 1         1
__ NAT0003      Invalid character string for file name or file n 1         1
__ NAT0004      DEFINE DATA must be the first statement if prese 1         1
__ NAT0005      Closing parenthesis missing in arithm/logical ex 1         1
__ NAT0006      ESCAPE statement used when no processing loop ac 1         1
__ NAT0007      Invalid THRU or TO clause in READ LOGICAL or HIS 1         1
__ NAT0008      Invalid search syntax.                           1         1
__ NAT0009      Invalid relational operator in a relational expr 1         1
__ NAT0010      Error in value specification in a relational exp 1         1
__
__
__
__
__
__

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

3 In the **Se** column, enter one of the following line commands:

| Command | Explanation |
|---|---|
| DE | Deletes the message. |
| DI | Converts the **Select Messages** screen to the **Display Short Messages** screen shown in *Displaying Messages*. Additionally, places the message selected with this command at the top of the list and reduces the number of messages displayed as described for `.X`. |
| LA | Shows the codes of the languages for which translations exist. |
| MO | Modifies the message. |
| PR | Outputs the message on a printer. |
| SH | Displays the short message.<br><br>This command is only available if an asterisk (*) has been entered in the **Language codes** field of the SYSERR main menu. |
| TR | Translates the message into another language. |

| Command | Explanation |
|---|---|
| .X | Defines a shorter message range by placing a selected message at the top of the list and thus reducing the number of messages displayed:<br><br>The message selected with this command is placed at the top of the list and any messages that were listed above this message are removed from the display. The message range in the SYSERR main menu is reset accordingly and starts with the message selected here on the **Select Messages** screen. |
| .Y | Defines a shorter message range by listing messages only up to a selected message:<br><br>All messages that were listed below the message selected with this command are removed from the display. The message range in the SYSERR main menu is reset accordingly and ends with the message selected here on the **Select Messages** screen. |

4 Press ENTER to continue.

## Translating Messages into other Languages

This function is used to translate short messages from one language to one or more other languages. To translate long messages into other languages, proceed as described in *Adding Messages*.

### To translate short messages

1 In the fields of the SYSERR main menu, enter the following values:

| Field | Input Value |
|---|---|
| **Code** | TR |
| **Message type** | NS Natural system short messages<br>US User-defined short messages |
| **Library** | Any existing Natural library. |
| **Message number** | Two numbers of up to four digits corresponding to the first and last numbers of the range of messages to be displayed for selection. |
| **Language codes** | Specify a maximum of nine language codes. The language codes are single alphanumeric characters in the ranges 1 - 9, A - Z and a - y. |

2 Press ENTER.

A **Translate Short Message** screen similar to the example below appears:

```
13:37:14             ***** NATURAL SYSERR UTILITY *****             2009-01-16
Lower Case ON            - Translate Short Message -

Number ...... SYSERR0001
Languages ... 1..45.....................................................

------------- ....+....1....+....2....+....3....+....4....+....5....+..
English       Short message English (1)_________________________________
German        ________________________________________________________________
French        ________________________________________________________________
Spanish       Short message Spanish (4)_________________________________
Italian       Short message Italian (5)_________________________________
              ________________________________________________________________
              ________________________________________________________________
              ________________________________________________________________
              ________________________________________________________________
------------- ....+....1....+....2....+....3....+....4....+....5....+..

 1 Short message English (1)
 4 Explanation: English long message
18 Action: English long message




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

The **Languages** field displays the language codes in which the message already exists (in the example above: 1, 4 and 5).

The section below **Number** and **Languages** lists all languages for which a language code was entered earlier in the **Language codes** field of the SYSERR main menu (in the example above: 1, 2, 3, 4, 5). English (1), Spanish (4) and Italian (5) translations already exist whereas new translations can be entered for German (2) and French (3).

For reference purposes, the bottom section of the screen displays three lines of the long message that corresponds to the language that is listed first in the languages/short messages section (in the example above, English). Lines 1, 4 and 18 are displayed by default. You can display any other line of the long message by overwriting any of the three line numbers (**1**, **4** or **18**) with another line number and pressing ENTER.

3 Enter the translation in the input line next to the new language specified.

4 Press ENTER.

### To modify translations of short messages

1 On the **Translate Short Message** screen, press PF10.

An **Options** window similar to the example below opens:

```
+------------------------ Options ------------------------+
!                                                           !
!  Modification of all fields allowed ....... N            !
!                                                           !
!  Currently recognized language codes ...... 123456789    !
!                                                           !
+-----------------------------------------------------------+
```

2 In the upper field, replace `N` (default) by `Y` and enter the required language code(s) in the lower field. You can specify up to nine new language codes for translation.

### To copy a translation into an empty input line

1 On the **Translate Short Message** screen, enter `.C` in the first two positions of an empty line.

2 Place the cursor anywhere in the line of a short message that already exists for another language. (You can only copy text that appears in display mode.)

3 Press ENTER.

# 75 Parameters


- Message Type ..... 506
- Library ..... 506
- Message Number ..... 506
- Language Codes ..... 506

This section describes the parameters that can be specified for a function in the SYSERR main menu. Any restrictions that apply to the use of parameters with a particular function are described in the section ***Functions***.

## Message Type

Specifies the type of message to be processed. The table below lists the message types available:

| Type | Explanation |
|---|---|
| NS | Natural system short messages |
| NL | Natural system long messages |
| US | User-defined short messages |
| UL | User-defined long messages |

## Library

Specifies the library for which messages are to be created or maintained. The specification of a library is not required when accessing Natural system messages (Message types NS and NL); any input values in the **Library** field are ignored.

## Message Number

Specifies the first and last number of a message range. The maximum message number for a library and language is 9999. The message number 0000 is not allowed. To specify only one message number, either enter the number of the message in the left **Message number** field and clear the right field, or enter the number in both fields.

## Language Codes

Specifies a maximum of 9 from 60 language codes available. The language codes are single alphanumeric characters in the ranges 1 - 9, A - Z and a - y. To view or select language codes, enter a question mark (?) in the first position of the **Language codes** field and press ENTER. For more information, see the system variable *LANGUAGE in the *System Variables* documentation.

# 76 Direct Commands

From the SYSERR main menu, you can execute the following commands by entering them in the command line:

| Command | Explanation |
|---|---|
| LAYOUT | Specifies valid message ranges to categorize messages. Overlapping of ranges is possible. A new message can only be added if its number is within the range specified in the layout. |
| NEXT | Searches for the next free message number within the message number range specified. Free means that this message number is available and has not yet been assigned to a message file in any language. |
| NEXTTAB | Same as NEXT, but returns a list of message numbers from which you can select a number. |
| RESTART | Re-initializes SYSERR (and its default values) without leaving the utility. |
| SAMPLE | Invokes the **Edit SAMPLE message** window where you create or modify a sample message to be used as a master for creating new short messages.<br><br>To create or modify a sample message, proceed as follows:<br><br>■ In the editor area of the **Edit SAMPLE message** window, type in the message text required or modify the existing text. If you enter the string 0000 (combined with text or not), the string 0000 is replaced by the number of the new message when copying the message. See also **Step 4** of *Adding Messages* in the section *Functions*.<br>■ In the **Read or Write sample** field, enter a W to save your entries.<br>■ In the **Library** field, enter the name of the library for which the sample message is to be used. If you leave the **Library** field blank, the sample applies to Natural system messages.<br>■ Press PF3 to exit the **Edit SAMPLE message** window.<br><br>You can define one sample message for each language and library. |
| SECURITY | Invokes a security window where you can enter a password and cipher code for accessing security-protected Adabas and VSAM files. |
| SHIFT | If activated, automatically shifts the text of a short message to the left margin when confirming a modification or adding a new message. |

| Command | Explanation |
| --- | --- |
| TRACE | Counts the number of database accesses. When the message number specified has been reached, a window is displayed. The default number is 900. If set to 0, the trace facility is shut off. The commands TRACE ON and TRACE OFF can be entered directly in the command line. TRACE ON sets the access counter to 900; TRACE OFF sets the access counter to 0. |
| USEREXIT | Invokes the USEREXIT program in the Natural system library SYSERR. |

# 77 Upper Case Conversion - ERRUPPER

Natural system messages are provided in lower case. If your terminals cannot display lower case characters correctly, convert the messages from lower to upper case by executing the program ERRUPPER in the Natural system library SYSERR.

However, once the messages have been converted to upper case, you cannot convert them back to lower case. To recover lower case messages, you have two options:

- Reload the messages by using the ERRLODUS program or the Object Handler.
- Unload the lower case messages to a free language code by using the ERRULDUS program or the Object Handler before conversion so that a backup always exists.

For detailed information, see ***Unloading Messages - ERRULDUS*** and ***Loading Messages - ERRLODUS***, and the *Object Handler* documentation.

# 78 Replacing Characters - ERRCHAR

If your terminal does not display certain characters correctly, it is possible to search for these characters and replace them by new characters of your choice. This is done by executing the program ERRCHAR in the Natural system library SYSERR. However, it is only possible to replace characters in Natural system short messages. When using ERRCHAR, you scan for a specific character and replace the hexadecimal code that represents this character with another hexadecimal code.

After executing the program ERRCHAR, the **ERRCHAR** menu is displayed with the following functions:

- Scan for a given character
- Scan and Replace characters
- Display one message in hexadecimal format
- EBCDIC character table for your terminal
- Translate using character set ERRCSET

The following input fields are provided in the **ERRCHAR** menu:

| Field | Explanation |
|---|---|
| **Message Number** | The range of messages to be included in the search or search/replace operation. |
| **Language Code** | The language code of Natural system short messages to be included in the search or search/replace operation. |
| **Scan Value** | The hexadecimal value to be scanned for. |
| **Replace Value** | The hexadecimal value to replace all scan values found. Use the function **EBCDIC character table for your terminal** to determine which characters your terminal can represent. |

# 79 Managing Messages in Different Libraries


- Unloading Messages - ERRULDUS ..... 514
- Loading Messages - ERRLODUS ..... 516

You can transfer messages between different libraries and rename, find, list or delete messages in different libraries. Depending on the Natural system environment(s) affected, you can either unload and load the messages to work files, or directly copy/move them from one library to another.

### To copy/move, rename, find, list or delete messages

- For libraries contained in the current Natural environment located on a mainframe platform, use the **SYSMAIN** utility as described in the relevant documentation.

  Or:

  For libraries contained in different Natural environments located on different mainframe, UNIX, OpenVMS or Windows platforms, use the *Object Handler* as described in the relevant documentation.

### To unload or load messages from or to a work file

- For libraries contained in a local mainframe environment, use the ERRULDUS and ERRLODUS programs described in the following sections.

  Or:

  For libraries contained in different Natural environments located on different mainframe, UNIX, OpenVMS or Windows platforms, use the *Object Handler* as described in the relevant documentation.

## Unloading Messages - ERRULDUS

The ERRULDUS program is used to unload all message types supported by Natural; that is, user-defined long and short messages created with the SYSERR utility and Natural system messages. The messages are read from the FNAT or FUSER system file and written into work file 2.

### To invoke ERRULDUS

- In the command line of the SYSERR main menu, enter `ERRULDUS`.

  An **ERRULDUS** menu similar to the example below is displayed:

```
16:11:13              ***** NATURAL SYSERR UTILITY *****              08-09-18
                   - ERRULDUS (Unload Texts to Work File 2) -

                     Code  Function
                     ----  ----------------------------------
                      US   User supplied short error texts
                      UL   User supplied long  error texts
                      U    User supplied short AND long texts
                      NS   NATURAL short error texts
                      NL   NATURAL long  error texts
                      H    NATURAL help texts
                      .    Exit
                     ----  ----------------------------------
   Code ............ __
   Source Library .. SYSERR__     Source Language Code .. 01
   Target Library .. SYSERR__     Target Language Code .. 01
   Error Number .... 1___ - 9999
   Replace ......... N
   Report .......... ON_

Please enter valid code.
Command ===>
Enter-PF1---PF2---PF3---PF4---PF5---PF6---PF7---PF8---PF9---PF10--PF11--PF12---
Exec  Help        Exit                                                   Canc
```

The **ERRULDUS** menu contains the following input fields:

| Field | Explanation |
|---|---|
| **Code** | The type of messages to be unloaded. Valid types are:<br><br>US User-defined short messages<br>UL User-defined long messages<br>U User-defined short and long messages<br>NS Natural system short messages<br>NL Natural system long messages<br>H Natural help texts<br>. Terminate processing |
| **Source Library** | The name of the library from which messages are to be unloaded. The name can be truncated by using an asterisk (*). If you enter an asterisk (*) only, messages from all libraries are unloaded. **Source Library** is ignored for messages of type NS or NL. |
| **Source Language Code** | The language code of the messages to be unloaded. |
| **Target Library** | The name of the library to which messages are to be loaded. **Target Library** is ignored for messages of type NS or NL. |
| **Target Language Code** | The language code to which messages are to be loaded. |

| Field | Explanation |
|---|---|
| **Error Number** | Two numbers of up to four digits corresponding to the first and last numbers of the range of messages to be unloaded. If no first number is specified, it is assumed to be 1. |
| **Replace** | Enter `Y` to overwrite the target library. Default is `N` (No). |
| **Report** | If specified as `ON`, a list of all messages that were unloaded is displayed when unloading is complete. If specified as `OFF`, no such list is displayed. |

ERRULDUS reads input until a period (.) is detected in the **Code** field. After unloading, a statistical listing is displayed.

## Loading Messages - ERRLODUS

The ERRLODUS program is used to load messages previously unloaded with the ERRULDUS program.

The messages are read from work file 2 and are written to the FNAT or FUSER system file. ERRLODUS overwrites existing messages in the system file if **Replace** was specified as `Y` when unloading.

If you specify ERRLODUS as `ON`, a list of all messages that were loaded (added or replaced) is displayed when loading is complete.

**Note:** Under Natural Security, the online use of ERRLODUS is only available for users of type Administrator, regardless of the setting of the **Utilities** option in the security profile of the library. See also Utilities in *Library Maintenance* in the *Natural Security* documentation.

# 80 Application Programming Interface USR0020P

The application programming interface USR0020P in the Natural system library SYSEXT is provided to read messages from the FNAT or FUSER system file. Thus, it is possible, for example, to have long messages displayed in an application (as part of your own user-defined help system) without having to use the Natural system library SYSERR.

Log on to the Natural system library SYSEXT and, in the command line, enter the command `MENU`. In the list provided, mark the program USR0020P with a question mark (?). A window is then displayed, in which you can select the function to be executed for the program. If you enter an `I`, further information on the use of USR0020P is displayed.

# XV SYSEXT Utility - Natural Application Programming Interfaces

# 81 SYSEXT Utility - Natural Application Programming Interfaces


- Introduction to SYSEXT ..... 522
- Invoking and Terminating SYSEXT ..... 524
- Using the SYSEXT Utility ..... 526
- Interface Versions ..... 531
- Reserved Keywords ..... 531
- Using a Natural API ..... 532

The utility SYSEXT is used to locate and test Natural Application Programming Interfaces (APIs) contained in the current system library SYSEXT.

A Natural API is a Natural subprogram (cataloged object) that is used for accessing and possibly modifying data or performing services that are not accessible by Natural statements. Natural APIs refer to Natural, a subcomponent or a subproduct.

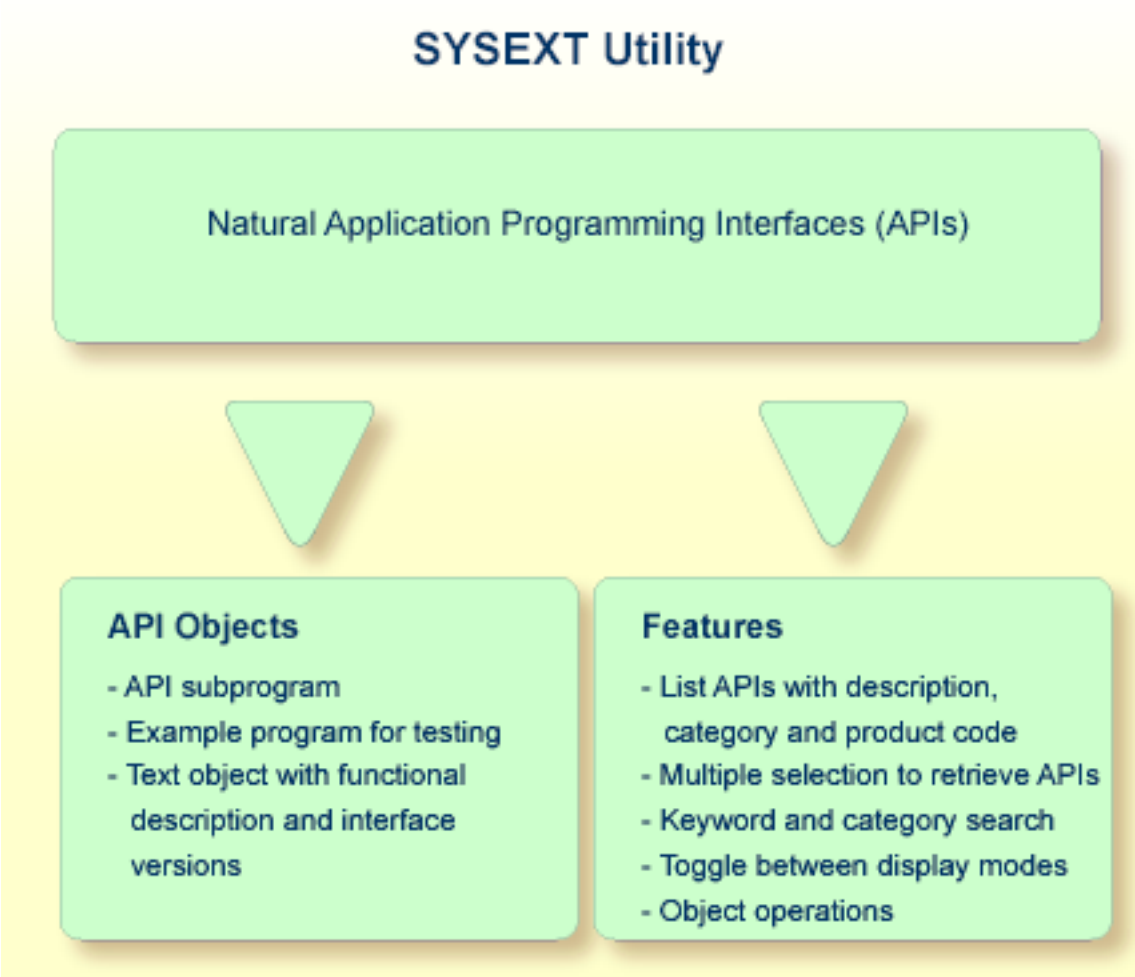
**Related Topics:**

- *Application Programming Interfaces - Natural Security* documentation
- *Application Programming Interfaces - Natural SAF Security* documentation
- *SYSAPI - APIs of Natural Add-on Products - Utilities* documentation

## Introduction to SYSEXT

For each Natural API, the utility SYSEXT provides a functional description, one example program, one category and API-specific keywords.

The following diagram is an overview of the Natural objects and major features SYSEXT provides:




### Objects Provided for Natural APIs

The types of Natural object typically provided for each Natural API are listed in the following section. Additional objects that might exist for a particular API are not covered.

All API-related objects are contained in the library SYSEXT on the system file FNAT.

In the following table, *nnnn* denotes the 4-digit number to identify the API as well as the corresponding example program and text object.

| Object Name | Explanation |
|---|---|
| USR*nnnn*N | The API subprogram (cataloged object) that performs the designated function. |
| USR*nnnn*P | An example program (source object) that can be used to test the effect of the API.<br><br>The example program invokes the corresponding subprogram USR*nnnn*N. |
| USR*nnnn*T | A text object listing a short and a long description, and information on usage, keywords, category and **interface versions**. |

| Object Name | Explanation |
|---|---|
| | You can display a text object by using the line command `T` as described in *Line Commands*. |

For some APIs, copycodes are available which provide functions related to the API. The copycodes are named `USRnnnnX`, where `X` is an identification character (such as "Z", "Y" etc.).

## Invoking and Terminating SYSEXT

The SYSEXT utility is invoked by the system command `SYSEXT` which is described by the following syntax diagram:

```
SYSEXT [ ALL     ] [ FIRST  ] [ DESCENDING ]
       [ CURRENT ] [ SECOND ] [ ASCENDING  ]
```

The table below gives a description of the parameters:

| Parameter | Description |
|---|---|
| ALL | List all APIs (default). |
| CURRENT | List only current APIs, that is all unique APIs and the current version of APIs with interface versions (with keyword +CURRENT-VERSION). See *Interface Versions*. |
| FIRST | Display first menu with product code items (default), see **below** for an example. |
| SECOND | Display second menu with category items, see **below** for an example. |
| ASCENDING | Display APIs in ascending order (default). |
| DESCENDING | Display APIs in descending order. |

### To invoke SYSEXT

- Enter the following system command:

  ```
  SYSEXT
  ```

  A menu similar to the example below appears with a list of all available Natural APIs displayed according to the first menu:

```
07:43:19             ***** NATURAL SYSEXT UTILITY *****            2010-09-15
User SAG                              - Menu -                   Library SYSEXT

Cmd Interface Description                                                Prod
--- USR*____  ____________________________________________________________ *__
 _  USR0010N  Get SYSPROF information                                     NAT
 _  USR0011N  Get information on logical file                             NAT
 _  USR0020N  Read any error message from FNAT or FUSER                   NAT
 _  USR0040N  Get type of last error                                      NAT
 _  USR0050N  Get SYSPROD information                                     NAT
 _  USR0060N  Copy LFILE definition from FNAT to FUSER                    NAT
 _  USR0070P  Define editor default profile 'SYSTEM'                      NAT
 _  USR0080N  Get or set type and name of editor contents                 NAT
 _  USR0100N  Maintain record length of VSAM datasets                     NVS
 _  USR0120N  Read Natural short error message                            NAT
 _  USR0210N  Save, catalog or stow Natural object                        NAT
 _  USR0220N  Read Natural long error message                             NAT
 _  USR0320N  Read user short error message from FNAT or FUSER             NAT

 Category .. ______________________________  Keyword .. ____________________

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

You can invoke SYSEXT with any parameters, for example to display the second menu at startup:

```
SYSEXT SECOND
```

This results in a menu similar to the one below:

```
07:43:19              ***** NATURAL SYSEXT UTILITY *****             2010-09-15
User SAG                             - Menu -                   Library SYSEXT

Cmd Interface Description                          Category
--- USR*____  ___________________________________  *______________________________
 _  USR0010N  Get SYSPROF information              SYSTEM COMMANDS
 _  USR0011N  Get information on logical file      SYSTEM FILES
 _  USR0020N  Read any error message from FNAT o   ERROR MESSAGES
 _  USR0040N  Get type of last error               ERROR HANDLING
 _  USR0050N  Get SYSPROD information              SYSTEM COMMANDS
 _  USR0060N  Copy LFILE definition from FNAT to   SYSTEM FILES
 _  USR0070P  Define editor default profile 'SYS   EDITOR
 _  USR0080N  Get or set type and name of editor   EDITOR
 _  USR0100N  Maintain record length of VSAM dat   VSAM
 _  USR0120N  Read Natural short error message     ERROR MESSAGES
 _  USR0210N  Save, catalog or stow Natural obje   NATURAL OBJECTS
 _  USR0220N  Read Natural long error message      ERROR MESSAGES
 _  USR0320N  Read user short error message from   ERROR MESSAGES

 Category .. ________________________________  Keyword .. ____________________

Command ===>
Enter-PF1---PF2---PF3---PF4---PF5---PF6---PF7---PF8---PF9---PF10--PF11--PF12---
      Help  Reset Exit  Desc  Curr  --    -     +     ++    <           Canc   ↩
```

As an alternative, you can invoke SYSEXT without parameters and then adjust the resulting menu, see *PF Keys*.

### To terminate SYSEXT

- On the SYSEXT utility menu, press PF3 or PF12.

  Or:

  In the command line, enter a period (`.`) or enter `EXIT`.

## Using the SYSEXT Utility

This section covers the following topics:

- Elements of the SYSEXT Utility Menu
- PF Keys
- Line Commands

- Utility Commands

### Elements of the SYSEXT Utility Menu

The SYSEXT utility menu provides a list of APIs. For each API, there is an API name (**Interface**), a description (**Description**), and, depending on the menu chosen, a classification according to product code (**Prod**) or category (**Category**). Except for the leftmost column (**Cmd**), each column is headed by a selection field that allows you to enter selection criteria.

You can also use the search field **Category** at the bottom of the menu to search for available categories.

The input field **Keywords** can either be used as search field to retrieve available keywords, or as selection field to retrieve APIs.

You can also specify selection criteria on multiple selection fields. For example, you can select in one step APIs beginning with `USR4`, and with keyword `PF-KEY` and category `NATURAL ENVIRONMENT`.

A detailed description of all menu elements and their usage is given below:

| Element | Explanation | Usage |
|---|---|---|
| **Cmd** | The input field for a line command to be executed on a text object or an example program: see *Line Commands*. | Enter a line command, example:<br><br>`K` List all keywords relevant to the specified API. |
| **Interface** | The name of the API subprogram. APIs with interface versions are displayed intensified. | Enter an asterisk (*) , or a prefix to be delimited by an asterisk, example:<br><br>`USR4*` List all APIs with prefix `USR4`. |
| **Description** | A brief description of the purpose of the API. | Enter a string, example:<br><br>`default` List all APIs with a description containing the string `default` or `Default`.<br><br>`*` List all APIs with a description that contains the string `*`. |
| **Prod** | The product code of Natural (`NAT`) or a Natural add-on product affected by the API.<br><br>Available product codes: `NAT` = Natural, `NCI` = Natural for CICS, `NDB` = Natural for DB2, `NVS` = Natural for VSAM, , `NDV` = Natural Development Server, `PRD` = Predict, `RPC` = Natural RPC (Remote Procedure Call). | Enter a name, or a prefix to be delimited by an asterisk (*), example:<br><br>`N*` List all APIs with a product code beginning with `N`. |

| Element | Explanation | Usage |
|---|---|---|
| | Product codes other than NAT are displayed intensified.<br><br>Only visible on the first menu. | |
| **Category (selection field)** | The category by which the API is classified according to its functional area or purpose.<br><br>An API can only have one category.<br><br>Only visible on the second menu and positioned above the list of APIs. | Enter a name, or a prefix to be delimited by an asterisk (*), example:<br><br>NATURAL OBJECTS List all APIs with category NATURAL OBJECTS. |
| **Category (search field )** | List all categories.<br><br>Positioned below the list of APIs. | See *Category Search*. |
| **Keyword** | List all keywords or enter a keyword as selection criterion. | See *Keyword Search* or *Keyword Selection*. |
| **Command** | Command line to enter utility commands. | Enter a utility command. See *Utility Commands*. |

### Category Search

- Enter an asterisk (*), or a prefix in the category search field optionally delimited by an asterisk, for example:

```
N*
```

As a result, a menu similar to the example below appears:

```
        Search for Categories

 Mark Category
 ---- N*____________________________
   _  NATURAL ENVIRONMENT
   _  NATURAL OBJECTS
```

To select a specific category as selection criterion, enter any character in the **Mark** column. As a result, a list of all APIs with this selection criterion is displayed.

### Keyword Search

- Enter an asterisk (*), or a prefix delimited by an asterisk, for example:

```
PF*
```

A menu similar to the example below appears with a separate window displaying keywords starting with PF.

```
        Search for Keywords

Mark Keyword
---- PF*_________________
  _  PF-KEY
  _  PF-KEY LINE
```

To select a specific keyword as selection criterion, enter any character in the **Mark** column. As a result, a list of all APIs with this selection criterion is displayed.

### Keyword Selection

- Enter a keyword in full, for example:

```
PF-KEY
```

As a result, a list of all APIs with keyword `PF-KEY` is displayed.

**Note:** A non-trailing asterisk is interpreted literally, for example entering `*LANGUAGE` results in a list of APIs with keyword `*LANGUAGE`.

## PF Keys

You can use the following PF keys:

| PF Key | Name | Function |
|---|---|---|
| PF1 | **Help** | Display context-sensitive help. There is a specific help text for each input field. In other contexts, for example the command line, a general help text is displayed. |
| PF2 | **Reset** | Clear all selection fields and readjust the list of APIs. |
| PF3 | **Exit** | Exit the SYSEXT utility, or the current menu or window. |
| PF4 | **Asc/Desc** | Toggle between ascending (**Asc**) and descending (**Desc**) order of APIs. |
| PF5 | **All/Curr** | Toggle between menus displaying all APIs (**All**) and menus displaying only current APIs (**Curr**). See *Interface Versions*. |
| PF6 | **--** | Scroll to the beginning of the list. |
| PF7 | **-** | Scroll one page up. |
| PF8 | **+** | Scroll one page down. |
| PF9 | **++** | Scroll to the end of the list. |
| PF10 | **<** | Shift to first menu. |
| PF11 | **>** | Shift to second menu. |
| PF12 | **Canc** | Exit the SYSEXT utility or the current menu or window. |

### Line Commands

Line commands are used to perform object operations. You can enter a line command in the **Cmd** column next to the API required. For a list of valid line commands, enter a question mark (?) or press PF1.

The following line commands are available:

| Line Command | Function |
|---|---|
| K | List keywords relevant to the specified API. |
| T | List text object USR*nnnn*T for a description of the corresponding API.<br><br>The description comprises purpose, function and calling conventions of the API, relevant keywords and its category. |
| L | List example program USR*nnnn*P. |
| E | Edit example program USR*nnnn*P. |
| R | Run example program USR*nnnn*P. |
| X | Execute example program USR*nnnn*P. |
| . | Exit the SYSEXT utility. |

### Utility Commands

This section covers the following utility commands to be entered in the command line:

- **EXIT**
  Exit the SYSEXT utility.
- **REFRESH**
  Update API information using data from the objects contained in the current library SYSEXT. The REFRESH command is only required if an API description or a keyword has been modified or if a text object has been added or removed.

  Upon successful completion, there is a confirmation that the text modules EXT-XML1 and EXT-XML2 have been generated in library SYSEXT.

  **Note:** Do *not* modify the source objects EXT-XML1 and EXT-XML2. They are required for configuring the SYSEXT utility and intended for Software AG internal use only.

## Interface Versions

Interface versions can be seen as a collection of APIs with (almost) the same functionality but with differently extended parameter specifications. Thus, they cover a development cycle to be kept explicit for sake of compatibility (of later versions with earlier versions).

If an API has interface versions, they are displayed in the corresponding text object USR*nnnn*T. Interface versions are ordered within a list according to the version they belong to. The rightmost element belongs to the current version. This status is expressed by the reserved keyword `+CURRENT-VERSION`. All other elements belong to a previous version and are marked with the reserved keyword `+PREVIOUS-VERSION`. APIs without interface versions are called unique.

APIs with interface versions are displayed intensified on the menu.

The list of all current APIs (see **PF5**) consists of all unique APIs and the current version of APIs with interface versions (with keyword `+CURRENT-VERSION`).

## Reserved Keywords

Reserved keywords refer to meta information on APIs, for example the Natural version in which an API has been added. Reserved keywords always start with a plus sign (+). See the table below for a description:

| Reserved Keyword | Description |
|---|---|
| +CURRENT-VERSION | The current version of an API with interface versions (see ***Interface Versions***). |
| +PREVIOUS-VERSION | A previous version of an API with interface versions (see ***Interface Versions***). |
| +NEW-*PROD-version* | An API that has been added to a specific product in a specific version. For example, `+NEW-NAT-4.2.7` refers to an API that has been added to the product Natural in version `4.2.7`. |
| +MOD-*PROD-version* | An API that belongs to a specific product and has been modified in a specific version. For example, `+MOD-NAT-8.2.1` refers to an API that belongs to product Natural and has been modified in version `8.2.1`. |

# Using a Natural API

If you want to use a Natural API contained in the system library SYSEXT, perform one of the following steps:

- Define the system library SYSEXT in the system file FNAT as a steplib library for the user library that contains the Natural objects that use this API. Thus, no API-specific actions are required when upgrading your Natural version.
- Copy the required API to the system library SYSTEM in the system file FNAT. Thus, you only need to check a single library for APIs when upgrading your Natural version.
- Copy the required API to the system library SYSTEM in the system file FUSER (not recommended).
- Copy the required API to the user library (or one of its steplibs) in the system file FUSER which contains the Natural objects that use this API (not recommended).

An API can only be used in the Natural version with which it is delivered. It is strongly recommended to store the APIs only in the FNAT system file. This will ensure that the right version is always executed.

### To make use of an interface

1. In the calling program, use the `DEFINE DATA` statement to specify the parameters listed in the text object `USRnnnnT` of that API. In the example program `USRnnnnP`, the parameters are defined within the `DEFINE DATA LOCAL` statement. Alternatively, you can specify the parameters outside the calling program in a separate LDA (Local Data Area) or PDA (Parameter Data Area), with a `DEFINE DATA LOCAL USING` statement referencing that data area.
2. Enter the following statement:

   ```
   CALLNAT 'USRnnnnN' parameters
   ```

   For further information, see the *CALLNAT* statement in the *Statements* documentation.

**Note:** Non-standard usage is always documented in the respective text object `USRnnnnT`.

If you want to use a Natural copycode contained in the system library SYSEXT, perform the following step:

- Copy the required copycode to the user library in the system file FUSER which contains the Natural objects that use this copycode.

### To make use of a copycode

1. In the calling program, use the `INCLUDE` statement to specify the parameters listed in the text object `USRnnnnT` of that API or in the copycode itself. In the example program `USRnnnnP`, the parameters are defined within the `DEFINE DATA LOCAL` statement. Alternatively, you can specify the parameters outside the calling program in a separate LDA (Local Data Area) or PDA (Parameter Data Area), with a `DEFINE DATA LOCAL USING` statement referencing that data area.
2. Some copycodes require additional data definitions. These are described in the text object `USRnnnnT` of the API and in the copycode itself.
3. Enter the following statement:

   ```
   INCLUDE USRnnnnX 'parameter'...
   ```

   For further information, see the *INCLUDE* statement in the *Statements* documentation.

**Note:** Non-standard usage is always documented in the respective text object `USRnnnnT`.

# XVI SYSEXV Utility

# 82 SYSEXV Utility


- Executing Example Programs of Current Versions ..... 538
- Executing Example Programs of Non-current Versions ..... 538
- Terminating the SYSEXV Utility ..... 539

The utility SYSEXV gives you access to examples of new features available in the current and in some earlier versions of Natural. All programs are available as source objects and display a detailed functionality description when they are executed. Example programs for non-current Natural versions are also available.

This chapter covers the following topics:

## Executing Example Programs of Current Versions

### To execute an example program of a current version

1 Enter the following system command:

```
SYSEXV
```

Or:

Log on to the SYSEXV library:

```
LOGON SYSEXV
```

and continue by entering

```
VERSION
```

As a result, a menu is displayed from which you can choose a Natural version.

2 Choose a version from the menu.

As a result, a commented menu is displayed from which you can choose an example program relevant to the selected version.

3 Choose a program.

As a result, the program is executed.

## Executing Example Programs of Non-current Versions

### To execute an example program of a non-current version

1 Log on to the SYSEXV library:

```
LOGON SYSEXV
```

2 List the text object `A-README`.

**Note:** Do not modify the text object `A-README`.

`A-README` lists and comments example programs for non-current versions.

3 Choose a program and execute it.

## Terminating the SYSEXV Utility

### To terminate the SYSEXV utility

- In the SYSEXV utility menu, choose **Exit**.

  Or:

  Press PF3 or PF12.

# XVII SYSMAIN Utility - Object Maintenance

The SYSMAIN utility is used to perform object maintenance functions such as copy, move, replace and delete.


General Information on SYSMAIN
Invoking and Terminating SYSMAIN
Using Menu Functions and Commands
Processing Programming Objects
Processing Debug Environments
Processing Error Messages
Processing Profiles
Processing Rules
Processing DL/I Subfiles
Processing DDMs
Processing Predict Sets
Keywords and Variables in Direct Commands
Special Commands Issued to SYSMAIN
Processing Status and Error Notification
Special Considerations for Administrators

# 83 General Information on SYSMAIN

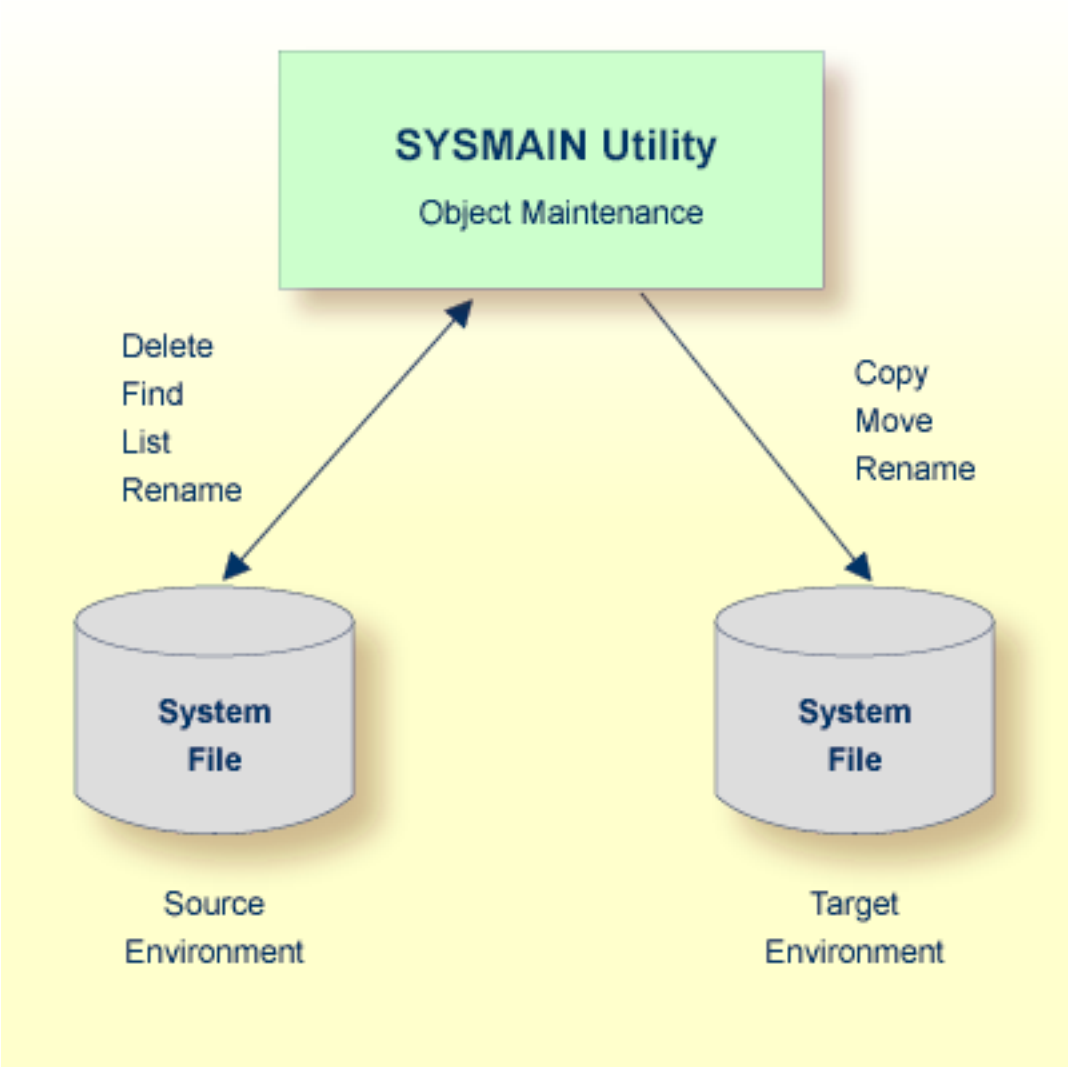

- Basic SYSMAIN Functionality ..... 544
- Object Types and Storage Location ..... 545
- Overview of Functions ..... 545

The SYSMAIN utility is used to maintain Natural objects in a Natural environment or across different environments.

This section provides basic information on object maintenance functions and the types of Natural object that can be processed with SYSMAIN.

## Basic SYSMAIN Functionality

The following diagram is a basic illustration of the SYSMAIN functionality:




The SYSMAIN utility copies or moves Natural objects from a source environment to a target environment and performs object operations such as delete in a source environment. The rename function can be performed in both a source and a target environment. An overview of the functions

and the Natural objects to which they apply is provided in the following section. The SYSMAIN utility functions are available online and in batch mode.

A source or target environment is an FNAT, FUSER or FDIC system file contained in an Adabas database or a VSAM file system. Natural objects on the FNAT or FUSER system file can be contained in libraries as indicated in the following section.

## Object Types and Storage Location

The types of Natural object that can be maintained with SYSMAIN are listed in the table below. The location of a Natural object depends on the object type as indicated in the table below:

| Objects/Subfiles | Description | Location |
|---|---|---|
| Programming objects | All Natural object types that are stored in a Natural library except debug environments and error messages. | Libraries in FUSER and FNAT |
| Debug environments | Debug environments for online program testing. | Library in FUSER |
| Error messages | Short and extended (long) Natural system error messages and user-defined error messages. | Libraries in FUSER and FNAT |
| Profiles | Editor profiles, map profiles, device profiles and parameter profiles. | FNAT |
| Rules | Automatic and free rules. | FDIC |
| DDMs | Data definition modules. | FDIC |
| DL/I subfiles | Natural NSBs, NDBs and UDFs. | FDIC |
| Predict sets | Retained sets created by the `LIST XREF` command or by Predict. Only available if Predict is installed. | FDIC |

## Overview of Functions

The functions provided by SYSMAIN are listed in the table below. The table also indicates which function is valid for each type of Natural object. For details on each function, refer to ***Description of Functions*** in *Using Menu Functions and Commands*.

| Function | Programming Object | Debug Environment | Error Message | Profile | Rule | DDM | DL/I Subfile | Predict Set |
|---|---|---|---|---|---|---|---|---|
| **Copy**<br>Copy object from one system file to another. | x | x | x | x | x | x | x | x |
| **Delete**<br>Delete object from a system file. | x | x | x | x | x | x | x | x |

| Function | Programming Object | Debug Environment | Error Message | Profile | Rule | DDM | DL/I Subfile | Predict Set |
|---|---|---|---|---|---|---|---|---|
| **Find**<br>Locate object in a system file. | x | | x | | | | | |
| **List**<br>Display a range of objects in a system file. | x | x | x | x | x | x | x | x |
| **Move**<br>Transfer object from one system file to another. | x | x | x | x | x | x | x | x |
| **Rename**<br>Give an object a new name or number (depending on its type), and (optionally) transfer it to a new system file. | x | x | x | x | x | | | x |

# 84 Invoking and Terminating SYSMAIN


- Invoking SYSMAIN Online or Batch ..... 548
- Invoking SYSMAIN with Appl. Programming Interface ..... 549
- Terminating SYSMAIN ..... 550

This section describes how to invoke and terminate the SYSMAIN utility online, in batch or by using an Application Programming Interface.

## Invoking SYSMAIN Online or Batch

The following instructions describe the methods of invoking the SYSMAIN utility by using a direct command (online or batch) or menu functions.

### To invoke SYSMAIN online

- From any library, enter the following Natural system command:

```
SYSMAIN
```

Or:

From the **Natural Main Menu**, invoke the **Maintenance and Transfer Utilities** menu and choose **Transfer Objects to Other Libraries**.

A **SYSMAIN Utility Main Menu** similar to the example below appears:

```
 12:41:50            ***** NATURAL SYSMAIN UTILITY *****           2013-04-05
 User SAG                        - Main Menu -

             Code  Object                                Code  Function

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

Object Code .. A                       Function Code .. _




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

The current setting of the system variable `*LIBRARY-ID` is passed to SYSMAIN and used as the default source library for processing programming objects and debug environments.

**To invoke SYSMAIN in batch**

- Use the following direct command:

```
SYSMAIN
```

followed by one or more command strings. See also *Issuing Direct Commands*.

## Invoking SYSMAIN with Appl. Programming Interface

MAINUSER is an Application Programming Interface (API) that can be used to perform SYSMAIN functions directly from any user-written object (for example, from a subroutine, program or subprogram) without going through the normal steps of invoking SYSMAIN.

Upon completion of processing of the SYSMAIN functions, the utility is terminated and control is returned to the program, subprogram or subroutine from which the request was issued.

MAINUSER is supplied as a cataloged object of the type subprogram in the system library SYSMAIN. MAINUSER can be used in either online or batch mode.

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

### Invoking MAINUSER

MAINUSER is invoked with the `CALLNAT` statement and its relevant parameters (see also *CALLNAT* in the *Natural Statements* documentation). MAINUSER must *not* be invoked from within the library SYSMAIN.

**To invoke MAINUSER**

- Issue a `CALLNAT` statement that contains the following syntax elements:

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

where the variable values denote the following parameters:

| Parameter | Natural Data Format/Length | Explanation |
|---|---|---|
| *command* | A250 | The direct command string to be executed by SYSMAIN. |
| *error* | N4 | The return code issued by SYSMAIN at the end of processing to indicate a normal end of processing or an error. |
| *message* | A72 | The message corresponding to the error given online. |
| *library* | A8 | The library containing SYSMAIN. If not specified, the default is SYSMAIN. |

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

## Terminating SYSMAIN

### To terminate SYSMAIN

- In the Command line of any SYSMAIN screen, enter one of the following direct commands:

```
.
```

(a period)

or

```
END
```

or

```
QUIT
```

Or:

Press PF3 (Exit), if required repeatedly.

Or:

In batch mode, use the either of the following direct command:

```
END
```

or

```
QUIT
```

See also ***Issuing Direct Commands*** in the section *Using Menu Functions and Commands*.

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

# 85 Using Menu Functions and Commands


- Performing Menu Functions ..... 552
- Executing Commands ..... 554
- Description of Functions ..... 556
- Function Processing and Reporting ..... 563
- SYSMAIN Online Help ..... 568

A SYSMAIN function can be performed in either menu or command mode.

This section describes how to use a SYSMAIN menu or a direct command and the options provided when performing a SYSMAIN function.

## Performing Menu Functions

In menu mode, you perform a SYSMAIN function by entering codes for the object type to be processed and the function to be performed. As an alternative to entering a function code, you can press a corresponding PF key.

### To perform a SYSMAIN menu function

- In the **SYSMAIN Utility Main Menu**, in the **Object Code** field, enter the one-letter code that corresponds to the objects required (in the example below, `A` for **Programming Objects**) and, in the **Function Code** field, enter the one-letter code that corresponds to the function required (in the example below, `C` for **Copy**). As an alternative to entering the function code, you can press the PF key that corresponds to the function (here: PF4); see also the list of **PF keys**.

```
 12:41:50            ***** NATURAL SYSMAIN UTILITY *****           2013-04-05
 User SAG                        - Main Menu -

             Code  Object                              Code  Function

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

Object Code .. A                       Function Code .. C




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

The appropriate object-type specific menu is then displayed as shown in the example of a **Copy Programming Objects** menu below:

```
17:28:52            ***** NATURAL SYSMAIN UTILITY *****           2005-08-10
User SAG                 - Copy Programming Objects -


                    Code  Function

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

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


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

The fields provided in an object-type specific menu depend on the SYSMAIN function performed. Fields that do not apply to a particular function are not displayed on the respective screen.

The fields are used to specify object selection criteria and processing options. For explanations of the fields, refer to the object-type specific sections of the SYSMAIN utility documentation.

## Using PF Keys

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

| PF Key | Name | Function |
|---|---|---|
| PF1 | Help | Display online help depending on the current cursor position.<br><br>If the cursor is positioned in the **Object Code** or **Function Code** field, SYSMAIN general help is displayed.<br><br>If the cursor is in another field, field-specific help is displayed. See also ***SYSMAIN Online Help***. |
| PF2 | Menu | Display the **SYSMAIN Utility Main Menu**. |
| PF3 | Exit | Return to the previous screen. If you press PF3 on the **SYSMAIN Utility Main Menu**, SYSMAIN is terminated. |

| PF Key | Name | Function |
|---|---|---|
| PF4 | Copy | Perform the **Copy** function for the object specified. |
| PF5 | Del | Perform the **Delete** function for the object specified. |
| PF6 | Find | Only applicable to programming objects and error messages.<br><br>Perform the **Find** function for the object specified. |
| PF7 | List | Perform the **List** function for the object specified. |
| PF8 | Move | Perform the **Move** function for the object specified. |
| PF9 | Ren | Not applicable to DDMs and DL/I subfiles.<br><br>Perform the **Rename** function for the object specified. |
| PF10 | Fsec | Invoke the security screen for specifying the Adabas security information of the FSEC system file if Natural Security is installed. |
| PF11 | Fdic | Not applicable to debug environments, error messages and profiles.<br><br>Only applies to programming objects, rules and DDMs.<br><br>Invoke the security screen for specifying the Adabas security information of the FDIC system file. If Predict is installed, you can also specify a Predict set user when performing the **Copy**, **Delete**, **Move** or **Rename** function on programming objects. |
| PF12 | Fnat | Not applicable to debug environments, error messages, Predict sets and profiles.<br><br>Invoke the security screen for specifying the system file information of the FNAT and/or FUSER system files. |

## Executing Commands

This section provides instructions for executing a SYSMAIN function by using a direct command or issuing a system command from within the SYSMAIN utility.

A direct command is used to perform a SYSMAIN function in either online or batch mode.

In batch mode, a report is automatically provided that shows the status of objects processed. This report can also be displayed in online mode: see ***Online Report Mode***.

- Issuing Direct Commands

- Using the SYSMAIN Command Line

## Issuing Direct Commands

Direct commands consist of a string of keywords that represent parameters. A period (.) indicates the end of a command. If this character is detected anywhere within a command string, all subsequent data is ignored.

The syntax that applies when issuing a direct command is described in the object-type specific sections of the SYSMAIN Utility documentation. The respective keywords and variables are explained in ***Keywords and Variables in Direct Commands***. Examples of direct commands are shown in the object-type specific sections, the SYSMAIN online help and on a **Selection** screen (see the **example screen** in *Selective Processing*).

### To issue a direct command online

- At any Natural command prompt, enter the following:

  ```
  SYSMAIN
  ```

  followed by a command string where each keyword can be delimited by a blank character instead of the delimiter.

  Or:

  In the Command line of any SYSMAIN menu, enter a direct command string where each keyword can be delimited by a blank character instead of the delimiter.

  Or:

  From within a Natural object, invoke the subprogram MAINUSER with the direct command string as a parameter where each keyword can be delimited by a blank character instead of the delimiter.

### To issue a direct command in batch

1 Use the command `SYSMAIN` and specify a command string in either of the following ways:

   1. The command string follows the `SYSMAIN` command in the same input line; each keyword in the command string can be delimited by a blank character instead of the delimiter.
   2. The command string follows the `SYSMAIN` command in the next input line; each keyword in the command string must be delimited by the delimiter and *not* by a blank character.

      If the direct command string is longer than one single line, the characters `CF` (see also the session parameter `CF` described in *CF - Character for Terminal Commands* in the *Parameter*

*Reference* documentation) must be placed at the end of the line to continue with the direct command in the next line.

2 If you want to execute other Natural commands after the `SYSMAIN` command(s), you must first terminate SYSMAIN by using the direct command `END` or `QUIT`.

### Using the SYSMAIN Command Line

In the Command line of any SYSMAIN menu, you can enter one of the following:

- A direct command for processing a SYSMAIN function.
- A special command to the SYSMAIN utility described in ***Special Commands Issued to SYSMAIN.***
- A system command. If the command is not uniquely identifiable as a system command, it must be preceded by two slashes (//). See also **Remarks** in *MAINEX05 - User Exit Routine for Verification of Direct Commands*.

## Description of Functions

The functions provided in a SYSMAIN menu or as direct commands are described in the following section. For each function provided in a menu, there is a corresponding direct command with the same name. Exception: **Help**.

For the syntax that applies when using a direct command, refer to the object-type specific sections of the *SYSMAIN Utility* documentation.

For the special commands that can be issued to the SYSMAIN utility, refer to ***Special Commands Issued to SYSMAIN***.

| Function/Command | Explanation |
|---|---|
| **Copy** | Copies a Natural object or a Predict set from a source environment to a target environment. The object remains unchanged in the source environment.<br><br>If the target environment already contains an object with the same name (or in the case of an error message or a Predict set, the same number) as the object to be copied, the specified object is not copied.<br><br>You can use the replace option (see ***Using the Replace Option***) to overwrite an object in the target environment. |

| Function/Command | Explanation |
|---|---|
| **Delete** | Deletes a Natural object or a Predict set from a source environment. During online **automated processing** (see *Function Processing and Reporting*), a confirmation window is displayed, which gives you the option of continuing or terminating the function.<br><br>For error messages: If Natural Security is installed, the delete function is disallowed for system error messages. When attempting to delete a system error message the following Natural system message occurs: `4897:Invalid error application specified`. |
| **Find** | Only applies to programming objects and error messages.<br><br>Locates one or more programming objects or error messages in a source environment.<br><br>During online processing, a window showing the library currently scanned is displayed.<br><br>In menu mode, the find function also provides the option to list and select libraries or reduce the number of objects displayed on a selection list: see ***Listing and Selecting Libraries*** or ***To shorten a selection list*** (*Using a Selection List*) respectively. |
| **List** | Displays a range of Natural objects or Predict sets in a source environment.<br><br>For programming objects, debug environments and error messages: In menu mode, the list function also provides the option to list and select libraries or reduce the number of objects displayed on a selection list: see ***Listing and Selecting Libraries*** or ***To shorten a selection list*** (*Using a Selection List*) respectively.<br><br>For programming objects: In batch mode, you can obtain a list of library names by using the direct command `LISTLIB`. |
| **Move** | Transfers a Natural object or a Predict set from a source environment to a target environment. The object is deleted from the source environment and added to the target environment. If the target environment already contains an object with the same name (or in the case of an error message, the same number) as the object to be moved, the specified object is not moved.<br><br>During online automated processing, a confirmation window is displayed, which gives you the option of continuing or terminating the function.<br><br>You can use the replace option (see ***Using the Replace Option***) to overwrite the object in the target environment. |
| **Rename** | The result of the rename function depends on the type of the called object:<br><br>**Type of Object** — **Result of Rename Function**<br>Error message — A new number is assigned.<br>Predict set — A new number and/or user and/or library is assigned.<br>DL/I subfile or DDM — Not applicable.<br>otherwise — The object is renamed.<br><br>Note that the function can be used in the following ways:<br><br>1. Perform the rename function on an object in the source environment. |

| Function/Command | Explanation |
|---|---|
| | 2. Perform the rename function on an object in the source environment and copy/move it to another (that is, target) environment. Note that the rename function can only be performed for a single environment at a time.<br><br>The rename function deletes the original object in the source environment; therefore, you are prompted with an option to retain the original object. If the original object is to be retained, it is not deleted.<br><br>If the target environment already contains an object with the same name (or in case of an error message or a Predict set, the same number) as the object specified, the rename function is not executed. In this case, you can use the replace option instead (see ***Using the Replace Option***) to overwrite the object in the target environment.<br><br>If automated processing is used, the rename function cannot process ranges of objects. Use **selective processing** (see below) instead.<br><br>For error messages: When renumbering a range of error messages within a single (source) library, the range values must not overlap. For example, it is not possible to rename error numbers 1 - 6 as new error numbers 5 - 10. A range of error messages can be renamed with **automated processing** (see below). When large ranges of error messages are being processed, the processing of messages may require significant resources. In such cases, batch-mode processing may be preferable. |
| **Help** | Provides help information on SYSMAIN: see ***SYSMAIN Online Help***.<br><br>The help function is only available in a SYSMAIN menu; there is no corresponding direct command. |
| **Exit** | Terminates the SYSMAIN utility.<br><br>PF3 (Exit) also terminates SYSMAIN if pressed in the **SYSMAIN Utility Main Menu**. |

This section covers the following topics:

- Using the Replace Option
- Listing and Selecting Libraries

## Using the Replace Option

If the target environment already contains an object with the same name as the object to be copied, moved or renamed, the specified object is not processed and processing continues with the next object. You can use the replace option to override this default feature and overwrite the object in the target environment.

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

#### To activate the replace option in command mode

- In the command string, specify the keyword `REPLACE`.

#### To activate the replace option in menu mode

1. In a SYSMAIN menu, in the **Replace** field, enter a `Y`.
2. Perform a function. If the **Sel. List** (Selection List) option has been set to `N`, a window appears where you can choose whether to confirm every replace operation before it is performed: Enter a `Y` to confirm each replacement, or press ENTER to continue processing without confirmation.
3. If you entered a `Y` to confirm a replacement, for each object that is to be replaced a window appears where you can enter one of the following characters:
   - `Y` replaces the object indicated in the window,
   - `N` does not replace the object indicated in the window (this is the default setting),
   - A period (.) terminates function processing. Alternatively, you can press PF3.

### Listing and Selecting Libraries

This option only applies to programming objects, debug environments and error messages.

When using the list function, you can invoke a selection list of libraries that contain the specified object(s). This also applies to the find function if the specified object is contained in several libraries.

#### To invoke a list of libraries for programming objects

- In a **List** menu, in the **Library** field, enter a range of library names (see also *Specifying a Range of Names*), and, in the **Name** field, enter an object name or a range of names.

  If you enter an asterisk (*) in the **Library** field (list function only), a list of all libraries available in the specified system file will be displayed.

  In the example of a **Library Selection** screen shown below, all libraries with names that start with `LIB` are selected:

```
 21:23:03             ***** NATURAL SYSMAIN UTILITY *****            2005-08-10
 User SAG                    -   Library Selection  -

LIST ALL * IN LIB* WHERE DBID 10 FNR 32

  C Library   S/C     C Library   S/C     C Library   S/C     C Library   S/C
  - --------  ---     - --------  ---     - --------  ---     - --------  ---
  _ LIB1      S/C     _ LIB2      S/C     _ LIB3        C     _ LIB4      S/C
  _ LIB5        S     _ LIB6      S/C     _ LIB7      S/C     _ LIB8      S/C
  _ LIB9      S/C     _ LIB10     S/C     _ LIB11     S/C     _ LIB12     S/C
  _ LIB13     S       _ LIB14     S/C     _ LIB15       C     _ LIB16     S/C
  _ LIB17     S/C     _ LIB18     S/C     _ LIB19     S/C     _ LIB20     S/C









                                                 Object Start Value: *

             Enter options (above), or '?' (Help) or '.' (Exit): _
Enter-PF1---PF2---PF3---PF4---PF5---PF6---PF7---PF8---PF9---PF10--PF11--PF12---
      Help  Menu  Exit  Copy  Del   Find  List  Move  Ren               Canc
```

The **Library** column lists all libraries that match the specified range. The **S/C** column indicates whether a library contains saved (source) objects and/or cataloged objects.

For a list of line commands available on the **Library Selection** screen, see *Selection Lists for Programming Objects*.

When using the find function, the **Library Selection** screen appears if the specified object is found in several libraries.

The **Library Selection** screen also appears when you leave a **Find Selection** or a **List Selection** screen by pressing PF3 (Quit).

### To invoke a list of libraries for debug environments

- In the **Debug Environments** menu:
  In the **Code** field, enter an L and, in the **Environment Name** field, enter the name of a debug environment or a range of names and, in the **Source Library** field, enter a range of library names (see also *Specifying a Range of Names*).

  If you enter an asterisk (*) in **Environment Name** and **Source Library**, a list of all libraries that contain debug environments in the specified system file will be displayed.

In the example of a **Library Selection** screen shown below, all libraries with names that start with `L` were selected:

```
 09:55:08            ***** NATURAL SYSMAIN UTILITY *****           2005-08-10
 User SAG                   -   Library Selection  -
LIST DEBUG * IN LIB* FROM DBID 10 FNR 32


  C Library           C Library           C Library           C Library
  - --------          - --------          - --------          - --------
  _ LIB1              _ LIB5              _ LIB9              _ LIB11
  _ LIB20 ...

                                          Object Start Value ... *
              Enter options (above), or '?' (Help) or '.' (Exit): _

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

The **Library** column lists all libraries that contain the specified debug environment(s). For a list of line commands available on the **Library Selection** screen, see *Selection Lists for Debug Environments*.

### To invoke a list of libraries for error messages

- Choose either of the following methods:

  1. In a **List** menu, in the **No. From** field, enter an error message number (in the example below, `1`) or specify a range of numbers by entering a start number in the **No. From** field and an end number in the **No. To** field.

     In the **Library** field, enter a range of library names (see also *Specifying a Range of Names*).

     A **Library Help** window similar to the example below appears:

```
14:02:47            ***** NATURAL SYSMAIN UTILITY *****          2005-08-10
User SAG                  - List Error Message Texts -
                              +---------------------------+
                              !  --- Library Help ---     !
                       Code  Fun !       - Source -        !
                              !                           !
                         A   Lis !   1   system messages   !
                         E   Lis !   2   LIB1              !
                         S   Lis !   3   LIB2              !
                         ?   Hel !   4   LIB3              !
                         .   Exi !   5   LIB4              !
                              !   6   LIB5                !
           Code ...... A      !   7   LIB6                !
                              !   8   LIB7                !
  Error    No. From .. 1___   !                           !
  Source   Library ... LIB*_  !                         ! 0___ FNR .. 32___
                              !   Enter selection or      !
                              !    '.' to Exit: 8_        !
                              +---------------------------+


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

From the window, select a library by entering the number that corresponds to the required library as shown in the example above where `8` was entered to select `LIB7`. If you have specified the correct FNAT system file, you can also select system error messages. If required, press enter to scroll to the end of the list.

2. In a **Find** menu, in the **Number** field enter an error message number, and, in the **Library** field, enter a range of library names (see also *Specifying a Range of Names*).

   A **Find Selection** screen similar to the example below appears for the specified range of libraries (here: `LIB*`):

```
 19:09:34            ***** NATURAL SYSMAIN UTILITY *****          2005-08-10
 User SAG                        - Find Selection -

FIND ERR 1 TYPE A FROM LIB* LANG * WHERE DBID 10 FNR 32


 C   Library   Error Message Text(s) for Error number: 1                Type
 -   --------  -------------------------------------------------------  ----
 _    LIB1      Short message of Error 1: wrong input value.              S
 _    LIB2      Short message of Error 1: incorrect syntax.               S
 _    LIB3      Short message of Error 1: undefined variable.             S
 _    LIB4      Short message of Error 1: timeout error.                  S
 _    LIB5      Short message of Error 1: initialization failed.         S/E
 _    LIB6      Short message of Error 1: invalid format.                S/E
 _    LIB7      Short message of Error 1: wrong password.                S/E
 _    LIB8      Short message of Error 1: input missing.                 S/E
 _    LIB9      Short message of Error 1: undefined keyword.             S/E
 _    LIB10     Short message of Error 1: invalid command.                S



        Enter options (above), or '?' (Help) or '.' (Exit): __
Enter-PF1---PF2---PF3---PF4---PF5---PF6---PF7---PF8---PF9---PF10--PF11--PF12---
      Help  Menu  Exit  Copy  Del         List  Move  Ren               Canc
```

The **Library** column lists all libraries that contain the specified error message(s) (here: `1`). The **Type** column indicates whether the short (`S`) and/or the extended (`E`) error message is available for the specified message.

For a list of line commands available on the **Find Selection** screen, see *Selection Lists for Error Messages*.

## Function Processing and Reporting

There are two types of processing that can be used when performing a SYSMAIN function: selective processing and automated processing.

This section describes the two types of function processing and how to obtain a status report about function processing:

- Automated Processing
- Selective Processing
- Using a Selection List

- Online Report Mode

### Automated Processing

Automated processing is the default type of processing when operating online in command mode. In batch mode, automated processing is the standard type of processing.

Automated processing is an online or batch facility which processes objects without displaying an intervening selection list. This requires little or no terminal I/O after a function has been selected.

Using automated processing online, the status of individual objects is not displayed, but an appropriate message is displayed upon completion of processing. If any of the following Natural system error messages is displayed, some or all of the specified objects were not processed:

```
4867:Nothing found for this request.
4810:All data rejected by these selection criteria.
4893:Normal completion, but some data were rejected.
```

The extended (long) message text of error NAT4810 lists reasons why an object may not have been processed. Reasons for an object not to be processed are also discussed in the section ***Processing Status and Error Notification***.

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

If required, when operating online, you can obtain a batch report with the processing status of all objects as described in ***Online Report Mode***.

### Selective Processing

Selective processing is the default type of processing when operating in menu mode.

Selective processing is an online facility which displays a selection list of all objects that meet the specified selection criteria. A selection list provides the following options:

- Select single or multiple objects for further processing.
- Perform an additional function on an object (for example, displaying the source code) before processing it (see: ***Using a Selection List***).
- Obtain individual messages on the processing status of each object (for example, `Moved` or `Copied`).
- View the direct command that corresponds to the input values entered in the SYSMAIN menu (see the **example of a selection screen** below). This may be of help when using direct commands instead of menu functions.

## Using a Selection List

This section provides instructions for invoking a selection list to further process single or multiple objects. In addition, this section describes how to shorten a selection when using the list or find function. This option is useful with large libraries in order to restrict the number of objects being displayed.

### To use a selection list online

1 In menu mode: In a SYSMAIN menu, in the **Sel. List** field, enter a `Y` (this is the default setting) to activate selective processing. Input of an `N` deactivates selective processing.

Or:

In command mode: Include the keyword `HELP` in the *`with-clause`* of the direct command or enter a question mark (?) immediately after the object name; see also the syntax diagrams in the object-type specific sections of the *SYSMAIN Utility* documentation.

When selective processing has been activated, a **Selection** screen similar to the example below appears:

```
 11:05:22            ***** NATURAL SYSMAIN UTILITY *****           2005-08-10
 User SAG                   -    Copy   Selection  -

COPY ALL * WITH XREF N FROM OLDLIB WHERE DBID 10 FNR 50 TO NEWLIB WHERE DBID
10 FNR 60

C  Name     Type   S/C  Message        C  Name     Type   S/C  Message
-  -------- ------ ---  ------------   -  -------- ------ ---  ------------
_  AA       Copycd S/C                 _  G0000001 Global S
_  G0000002 Global S/C                 _  G0000003 Global S/C
_  L0000001 Local  S/C                 _  L0000002 Local  S/C
_  L0000003 Local  S                   _  M0000001 Map    S
_  M0000002 Map    S/C                 _  M0000003 Map    S/C
_  P0000001 Progrm S                   _  P0000002 Progrm   C
_  P0000003 Progrm S/C                 _  P0000004 Progrm S/C
_  P0000005 Progrm S                   _  P0000006 Progrm S/C
_  P0000007 Progrm S/C                 _  S0000001 Subpgm S/C
_  S0000002 Subpgm S/C                 _  S0000003 Subpgm S
_  T0000001 Text   S                   _  T0000002 Text
                                                     Listed Library: OLDLIB

              Enter options (above), or '?' (Help) or '.' (Exit) _
Enter-PF1---PF2---PF3---PF4---PF5---PF6---PF7---PF8---PF9---PF10--PF11--PF12---
      Help  Menu  Exit  Copy  Del   Find  List  Move  Ren               Canc
```

The list contains the names and the types of the specified objects and indicates whether saved objects (`S`) and/or cataloged (`C`) objects exist.

The lines above the selection list (highlighted in the example above) displays the SYSMAIN direct command that corresponds to the input values you entered in the menu fields.

This command corresponds to the command syntax that applies when you work in command mode, although some keywords are optional, as shown in the syntax diagrams in the object-type specific sections of the SYSMAIN documentation.

2 Select one or more objects for further processing: In the **C** (command) column next to the object(s) desired, enter one of the line commands described in the object-type specific sections of the *SYSMAIN Utility* documentation.

3 Press ENTER to perform one or more line commands.

Line commands are executed in alphabetical order of the specified object names whereby commands that perform a SYSMAIN maintenance function (for example, delete) are always executed last.

If *no* SYSMAIN maintenance function (for example, displaying source code) was performed on any object, you can again enter a line command for any object(s). However, once a SYSMAIN maintenance function has been performed on any object(s), the fields of the **C** column are no longer available for input.

If a line command has been executed, the status of the object(s) processed is displayed in the **Message** column as shown in the example below:

```
 11:05:22             ***** NATURAL SYSMAIN UTILITY *****           2005-08-10
 User SAG                    -    Copy   Selection  -

COPY ALL * WITH XREF N FROM OLDLIB WHERE DBID 10 FNR 50 TO NEWLIB WHERE DBID
10 FNR 60

C  Name     Type   S/C  Message        C  Name     Type   S/C  Message
-  -------- ------ ---  ------------   -  -------- ------ ---  ------------
_  AA       Copycd S/C  Copied         _  G0000001 Global S    Copied
_  G0000002 Global S/C  Copied         _  G0000003 Global S/C  Copied
_  L0000001 Local  S/C  Copied         _  L0000002 Local  S/C  Copied
_  L0000003 Local  S    Replaced       _  M0000001 Map    S    Replaced
_  M0000002 Map    S/C  Replaced       _  M0000003 Map    S/C  Replaced
_  P0000001 Progrm S                   _  P0000002 Progrm   C
_  P0000003 Progrm S/C                 _  P0000004 Progrm S/C
_  P0000005 Progrm S                   _  P0000006 Progrm S/C
_  P0000007 Progrm S/C                 _  S0000001 Subpgm S/C
_  S0000002 Subpgm S/C                 _  S0000003 Subpgm S
_  T0000001 Text   S                   _  T0000002 Text
                                                     Listed Library: OLDLIB

              Enter options (above), or '?' (Help) or '.' (Exit) _
Enter-PF1---PF2---PF3---PF4---PF5---PF6---PF7---PF8---PF9---PF10--PF11--PF12---
      Help  Menu  Exit  Copy  Del   Find  List  Move  Ren               Canc
```

For a list of possible status messages, see ***Status Messages*** in the section *Processing Status and Error Notification*.

### To shorten a selection list

- On a **List Selection** or **Find Selection** screen, in the **Reposition to** field, enter the name of an object or specify a range of names (see ***Specifying a Range of Names***) and press ENTER.

  If you have specified a name, the list now starts from the specified name. If you have specified a range of names, the list now displays only objects within that range.

  The **Reposition to** option is only valid in online mode. It is not a selection criterion for the list function.

## Online Report Mode

Online report mode can be used to obtain a SYSMAIN batch report online instead of a selection list. An online batch report lists all objects that were affected by a SYSMAIN function and indicates the action performed on each of these objects.

### To use a batch report online

1. In a SYSMAIN menu, in the **Sel. List** (Selection List) field, enter an `N` to activate automated processing.
2. In the Command line, enter one of the following direct commands:

   ```
   BATCH
   ```

   or

   ```
   BAT
   ```

   A message appears confirming that batch mode has been activated.
3. If required, you can obtain a hardcopy of the report by entering the following terminal command:

   ```
   %H
   ```

   (See also *%H - Hardcopy Output* in the *Terminal Commands* documentation.)
4. Execute a SYSMAIN function. SYSMAIN now processes the function as if in batch mode. Hence, only the result of each action is present in a report-type format as shown in the example below:

```
10:50:30              ***** NATURAL SYSMAIN UTILITY *****           2005-08-10
User SAG                    - Copy   Function -

COPY ALL * WITH XREF N FROM TESTLIB1 WHERE DBID 10 FNR 30 TO TESTLIB2 WHERE
DBID 10 FNR 40




Saved     Progrm TEST1    has now been Copied
Saved     Progrm TEST2    has now been Copied
Saved     Progrm TEST3    has now been Copied
Stowed    Progrm TEST4    has now been Copied
Stowed    Progrm TEST5    has now been Copied
Stowed    Progrm TEST6    has now been Copied
Stowed    Progrm TEST7    has now been Copied
Saved     Record TEST8    has now been Copied
Saved     Text   TEST9    has now been Copied
Cataloged Progrm TEST10   has now been Copied
Saved     Progrm TEST11   has now been Copied
Stowed    Progrm TEST12   has now been Copied
MORE
Stowed    Progrm TEST13   has now been Copied
Cataloged Progrm TEST14   has now been Copied
Stowed    Progrm TEST15   has now been Copied
```

5 If required, you can interrupt function processing by entering a system command or terminal command at a MORE prompt. Otherwise, press ENTER until you have reached the end of the list and return to the SYSMAIN menu where a message confirms successful execution of the function.

## SYSMAIN Online Help

The SYSMAIN online help facility provides information on all functions provided by SYSMAIN including detailed explanations of the direct command syntax and examples of direct commands.

In addition to the help facility, SYSMAIN provides individual information on any input field available on any SYSMAIN screen.

### To invoke SYSMAIN help topics

1 In the **SYSMAIN Utility Main Menu**, position the cursor in the **Object Code** or the **Function Code** field and press PF1 (Help) or enter a question mark (?).

The **Help Menu** of the SYSMAIN utility appears similar to the example below with a list of help topics provided:

```
15:54:28            ***** NATURAL SYSMAIN UTILITY *****           2005-08-10
User SAG                       - Help Menu -
                                                            Help Name SHT-0001

                 Code  Topic

                   A   SYSMAIN General Overview
                   C   Command Mode
                   E   Environment Definition
                   F   Functions / Commands
                   S   Security Environment
                   .   Exit





          Code ... _

Select a function code.

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

2 In the **Code** field, enter the one-letter code that corresponds to the help topic desired.

3 Press ENTER.

A result screen appears with information on the help topic selected, or another menu is invoked with further help topics that help narrow down your search.

### To invoke help on an input field

- Position the cursor in the field in question and press PF1 (Help) or enter a question mark (?) and press ENTER.

  A window appears with field-specific instructions and, where applicable, a list of valid input values.

# 86 Processing Programming Objects


- Fields in Programming Objects Menus ..... 572
- Using Profile Parameter RECAT ..... 575
- Selection Lists for Programming Objects ..... 575
- XRef Considerations ..... 579
- Specifying Additional Criteria ..... 581
- Direct Command Syntax for Programming Objects ..... 583

All SYSMAIN functions can be performed on programming objects. Programming objects that can be maintained with SYSMAIN include the following types of Natural object: program, subprogram, subroutine, copycode, helproutine, map, local data area, global data area, parameter data area, class, text, recording, Natural command processor, dialog, function, ISPF macro, report, adapter and resource.

Programming objects are stored in the system files according to the name of the library in which they are contained: if the library begins with SYS (except for the library SYSTEM), objects are stored in the FNAT system file. In all other libraries, the objects are stored in the FUSER system file.

This section describes menu functions and selection list options provided to perform a SYSMAIN function on programming objects and the syntax that applies when using direct commands.

## Fields in Programming Objects Menus

The **Programming Objects** menus contain all SYSMAIN functions required for the processing of programming objects. The fields that can be provided in a menu are described in the following table:

| Field | Explanation | |
|---|---|---|
| **Recat: ON** | Indicates that the profile parameter `RECAT` has been set to `ON`: see ***Using Profile Parameter RECAT***. | |
| **Code** | Specifies whether a saved (source) object and/or a cataloged object is to be processed: | |
| | | |
| | `A` | Any object that exists as a saved object and/or a cataloged object is processed. |
| | `C` | Any object that exists as a cataloged object is processed. |
| | `S` | Any object that exists as a saved object is processed. |
| | `W` | Only an object that exists as both a saved object *and* a cataloged object is processed. The exceptions to this are copycode, text and recording which cannot be cataloged. However, they are included in processing when this option is specified. |
| **Sel. List** | Specifies whether selective processing or automated processing is used: | |
| | | |
| | `Y` | Yes.<br>Selective processing is activated. A selection list is displayed when processing objects. This is the default setting. For the columns, fields and line commands available on a selection list, see ***Selection Lists for Programming Objects***. |

| Field | Explanation |
|---|---|
| | N — No.<br>Selective processing is deactivated. |
| **Name** | The name of the object to be processed or a range of names: see also *Specifying a Range of Names*.<br><br>The default setting is an asterisk (*) which means that all names are selected. |
| **New Name** | The name to be given to an object when it is renamed with the rename function. |
| **Type** | The code that corresponds to the object type(s) to be processed such as P for program and M for map: see *TYPE Specification - Programming Objects*.<br><br>You can enter one or more codes in any sequence. For example, if you enter PAM, programs, parameter data areas and maps are processed. |
| **Set Number** | The number of the retained set created with the Predict XRef save set option of the LIST XREF command. You can apply all SYSMAIN processing functions to the objects included in this set.<br><br>If any valid number is specified, SYSMAIN assumes a Predict set. If no number is specified, normal object processing is assumed.<br><br>You can specify a library and a user ID for the Predict set by using the fields **Set Library** and **Set User** in the **Additional Criteria** window described in *Specifying Additional Criteria*. |
| **XREF** | Indicates whether XRef (cross-reference) data stored on Predict system files is to be processed:<br><br>N — No.<br>XRef data is not processed, except when using the delete function. If a cataloged object is deleted or replaced, SYSMAIN always deletes any existing XRef data for this object. N is the default setting.<br><br>Y — Yes.<br>All XRef data is processed.<br><br>S — Special.<br>A specified object is processed regardless of whether it has cross-reference data or not. Any existing XRef data is processed.<br><br>F — Force.<br>All XRef data is processed and the object must be documented in Predict.<br><br>D — Documented.<br>The object must be documented in Predict. Any existing XRef data is processed.<br><br>For further details, see *XRef Considerations*. |
| **Library** | The name of a source or a target library.<br><br>The source library contains the object to be processed. The target library is the library to which the object is to be copied or moved, or where the object is renamed. |

| Field | Explanation |
|---|---|
| | See also *Listing and Selecting Libraries*. |
| **Database** | The database ID (DBID) of a source or a target database.<br><br>The source database contains the library and system file where the object to be processed is stored. The target database contains the library and system file to which the object is to be copied or moved, or where the object is renamed.<br><br>Valid database IDs are `1` to `65535`. |
| **File** | The file number (FNR) of a source or a target system file (FNAT or FUSER).<br><br>Valid file numbers are `1` to `65535`.<br><br>The source file contains the library where the object to be processed is stored. The target file contains the library to which the object is to be copied or moved, or where the object is renamed. |
| **Replace** | Specifies whether an object is to be replaced when using the move, copy or rename function:<br><br>`Y` — Yes. An object with the same name which exists in the target environment is replaced.<br><br>`N` — No. An object with the same name which exists in the target environment is not replaced. This is the default setting.<br><br>See also *Using the Replace Option*. |
| **Criteria** | Invokes the **Additional Criteria** window where you can specify additional object selection criteria. Possible values are:<br><br>`N` — No. The **Additional Criteria** window is not invoked. This is the default setting.<br><br>`Y` — Yes. The **Additional Criteria** window is invoked.<br><br>See also *Specifying Additional Criteria*. |

## Using Profile Parameter RECAT

If the profile parameter `RECAT` has been set to `ON` at session start, this is indicated in the **Programming Objects** menus and on the **Selection** screens.

Using SYSMAIN functions with `RECAT=ON`, normal dynamic recatalog rules apply as described in *RECAT - Dynamic Recataloging* in the *Parameter Reference* documentation. This means the following:

- If an object exists as both a saved object and a cataloged object, neither the saved object nor the cataloged object can be processed independently.
- If an object only exists as a cataloged object, you cannot perform a SYSMAIN function that only processes cataloged objects (error message `Invalid request with dynamic recatalog` appears) or select a cataloged object from a **Selection** screen.

When automated processing is used, any object not satisfying these rules is ignored and processing continues with the next object.

## Selection Lists for Programming Objects

If selective processing has been activated, a selection list of all programming objects that meet the specified selection criteria is displayed on a **Selection** screen.

This section describes the columns and fields contained on a **Selection** screen and the line commands provided to further process a programming object:

- Columns and Fields
- Line Commands

### Columns and Fields

The following columns and fields are displayed on a **Selection** screen:

| Column/Field | Explanation |
|---|---|
| **C** | Input field for line commands (see below). |
| **Name** | The name of the programming object that meets the specified selection criteria. |
| **Type** | The code that corresponds to the type of object as listed in *TYPE Specification - Programming Objects*. |
| **S/C** | The object that exists for the programming object: a saved/source (`S`) object and/or a cataloged (`C`) object. |
| **Message** | The message that indicates the processing status of a programming object. For possible messages, see *Status Messages*. |

| Column/Field | Explanation |
|---|---|
| **Listed Library** | The name of the library that contains the selected programming object(s). |

## Line Commands

One of the following line commands can be entered in the **C** (Command) column of a **Selection** screen:

| Line Command | Function |
|---|---|
| A | Process any object listed in the **S/C** (Saved/Cataloged) column; that is, saved/source (S) objects and/or cataloged (C) objects.<br><br>This command does not apply to the find or the list function. |
| B | Delete the cataloged object from the Natural buffer pool. Deletion of the specified object(s) must be confirmed by entering DELETE in a window that appears once you have specified the object(s) and pressed ENTER. |
| C | Process only the cataloged object, even if there is a corresponding saved object. If C is specified for an object that exists only as a saved object, an error occurs.<br><br>This command does not apply to the find or the list function. |
| D | Only applies to a **Library Selection** screen (see also ***Listing and Selecting Libraries***).<br><br>Display a short list of the objects contained in the specified library. The information contained in the list (name, type, source/cataloged object) is identical to the information displayed on the **Selection screen** shown in *Selective Processing* in *Using Menu Functions and Commands*. |
| H | Produce a hardcopy of the saved (source) object.<br><br>The source code of the specified object is printed and displayed on the screen. |
| I | Display directory information of an object.<br><br>This command is similar to the system command LIST DIR *object-name*; for details on the directory information displayed, refer to *Displaying Directory Information* in the *System Commands* documentation. |
| L | Display the source code of a saved (source) object.<br><br>This command corresponds to the system command LIST *object-name*; for the commands that can be executed from a source-code screen, refer to *List of Source* in the *System Commands* documentation. Exceptions: The command EXPAND is not available. The command ZOOM cannot be used if the source object to be displayed is contained in a steplib library. |
| | On a **Library Selection** screen (see also ***Listing and Selecting Libraries***):<br><br>Display an extended list of the objects contained in the library. In addition to the information displayed by using line command D, the extended list provides information of the object directory: programming mode, Natural version, user ID, saved/cataloged date and time. |
| R | Display the long name of a Natural object of the type resource. |

| Line Command | Function |
|---|---|
| | On a **Library Selection** screen (see also *Listing and Selecting Libraries*):<br><br>Verify use of external subroutines: For the library and object range specified, display the objects for which a cataloged object exists and indicate whether an object references an external subroutine.<br><br>For each object that references an external subroutine, display the name of the external subroutine and the name of the cataloged object that exists for this subroutine if available. For an object of the type subroutine, the name of its equivalent alias long name is displayed. |
| S | Process only the saved (source) object, even if there is a corresponding cataloged object. If S is specified for an object which exists only as a cataloged object, an error occurs.<br><br>This command does not apply to the find or the list function. |
| | On a **Library Selection** screen (see also *Listing and Selecting Libraries*):<br><br>List all external subroutines contained in the specified library by their alias long names and/or the short names of their equivalent cataloged objects. |
| X | Only applies if Natural Connection and Entire Connection are installed.<br><br>Download saved (source) object(s) to a PC: see also *Downloading Source Objects to a PC*. |
| Z | Calculate sizes: review the sizes of the saved (source) and cataloged objects, for example, the DATSIZE, ESIZE and MCG size. |

### Downloading Source Objects to a PC

The download option only applies if Natural Connection and Entire Connection are installed.

#### To download one or more source objects

1 Before you invoke SYSMAIN:

- Define work files 6 and 7 as PC work files.
- After session start, activate the PC connection by entering the following terminal command:

```
%+
```

(See also *Enable/Disable Use of Natural Connection* in the *Terminal Commands* documentation).

2 Invoke any **Selection** screen.

3 Next to the object(s) you want to download, enter the following line command:

```
X
```

and press ENTER.

A **PC Download Options** window similar to the example below appears:

```
 13:23:15            ***** NATURAL SYSMAIN UTILITY *****           2005-08-10
 User SAG                    -    Copy   Selection  -

COPY ALL * WITH XREF N FROM OLDLIB WHERE DBID 10 FNR 32 TO NEWLIB WHERE
DBID 10 FNR 32

 C  Name     Type   S/C  Message        C  Name     Type   S/C  Message
 -  -------- ------ ---  ------------   -  -------- ------ ---  ------------
 X  SUB1     Subpgm S                   X  SUB2     Subpgm S/C
 X  SUB3     Subpgm S/C                 X  SUB4     Subpgm S
 X  SUB5     Subpgm S/C                 X  SUB6     Subpgm S/C
+----------------- PC Download Options ------------------+ S/C
!   Specify the relevant PC options                      ! S/C
!                                                        ! S
! Drive ...... C                                         ! S/C
! Path ....... SOURCES/SUBPROGRAMS                       ! S/C
! Extension .. NS*                                       ! S
!                                                        ! S
!                                                        ! S
!                                                        ! ed Library: OLDLIB
! Warning: This will overwrite any existing objects with !
!          the same path, name and extension.            ! Exit) _
+-------------------------------------------------------+ -PF10--PF11--PF12---
      Help  Menu  Exit  Copy  Del   Find  List  Move  Ren              Canc
```

In the **Drive** field, enter the name of the PC drive to which you want the object(s) to be downloaded. The default setting is `C`.

In the **Path** field, enter the name of the PC directory/subdirectory to which you want the object(s) to be downloaded (in the example above, subdirectory *SUBPROGRAMS* in directory *SOURCES*). Enter a slash (/) as the separator between a directory and subdirectory, and between subdirectories. If the specified directory/subdirectory does not exist, you will receive an appropriate error message.

In the **Extension** field, enter the extension of the text file into which the source code of the object is to be loaded. If you specify `NS*`, the asterisk (*) will be replaced by the specified object type (see ***TYPE Specification - Programming Objects***).

For example: a subprogram with the name SUB1 will be loaded into a file with the name *SUB1.NSN*.

4 Press ENTER to download the object(s).

The following message appears in the message line: `4824:Requested option(s) processed successfully`.

Additionally, if the **Message** column is displayed on the screen, the message `Exported` appears next to the object(s) downloaded.

5 If desired, continue downloading other objects from this selection list.

The **PC Download Options** window will not appear again for any further downloads from the current **Selection** screen. This window only appears when the line command `X` is issued for the first time after invoking a **Selection** screen, or after executing the direct command `SET PC`.

The settings in the **PC Download Options** window remain active until you terminate SYSMAIN.

## XRef Considerations

All cross-reference (XRef) data stored in the Predict system file can be processed with SYSMAIN. The XREF option indicates whether SYSMAIN should process XRef data. XRef data is always deleted if the delete or replace function is performed on a cataloged object.

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

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

Consider the following settings for XREF:

- XREF set to N
- XREF set to Y or F
- XREF set to F
- XREF set to S
- XREF set to D
- XREF Errors

### XREF set to N

If the XREF option is set to `N`, no XRef data is processed, but in situations where a cataloged programming object is deleted or replaced, SYSMAIN deletes the XRef data. The target Predict system file is determined according to the current settings of the source or target FDIC system file. The default is the value assigned to the profile parameter `FDIC` (see *FDIC - Predict System File* in the *Parameter Reference* documentation) at the start of the Natural session.

### XREF set to Y or F

If the XREF option is set to Y or F, the following actions are applied during processing:

- SYSMAIN verifies that XRef data already exists in the Predict system source file.
- If the replace option is active (set to Y) and a programming object is to be deleted from the target environment, XRef data is deleted from the Predict system target file.
- If a programming object is being copied to a new environment, the XRef data of the programming object is copied from the Predict system source file to the Predict system target file. The library name is changed accordingly and in the case of the rename function, the object name is also changed.
- If the move function was requested, the XRef data of the programming object is deleted from the Predict system source file.

### XREF set to F

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

### XREF set to S

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

### XREF set to D

If the XREF option is set to D, SYSMAIN checks that the programming object (program, subroutine, subprogram, map or helproutine only) has a Predict program entry defined on the Predict system target file. If not, processing of the object is terminated. Objects that have cross-reference data are processed with their cross-reference data, and the objects that have none are also processed.

### XREF Errors

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

- The value of the XREF option in Natural Security is F or Y and you specified a value of Y or N respectively.
- The XREF option is set to F or D, and SYSMAIN finds no documented program entry in Predict for the object being processed.

- An invalid Predict file is specified.
- The value of the XREF option in Natural Security is `F`, `D` or `Y` and you specified a value of `S`.
- The value of the XREF option in Natural Security is `D` and you specified a value of `N`.

## Specifying Additional Criteria

In addition to the selection criteria specified in the input fields in a **Programming Object** menu, you can select objects by a date/time, user ID and terminal ID that relates to their saving or cataloging.

You can also specify the user ID and the library for a Predict set. This option does not apply to the list and find functions.

For example, you can select only those objects that were cataloged on a specific day between 8:00 and 12:00 by a specific user on a specific terminal, which means that the processing of objects according to the selection criteria is based on all selected criteria as a whole, not on each condition.

### To specify additional selection criteria

1 In the **Criteria** field of a **Programming Objects** menu, replace `N` (default) by `Y`.

  The **Additional Criteria** window similar to the example below appears:

```
15:24:55            ***** NATURAL SYSMAIN UTILITY *****           2006-09-15
User MMO                 - Copy Programming Objects -
                  +-----------------------------------------+
                  !       --- Additional Criteria ---        !
                  !                                          !
                  !  Object Type ..... PN_________________   !
                  !  Date/Time From .. 2006-01-01 _____      !
                  !  Date/Time To .... __________ _____      !
                  !  User ID ......... SAG______             !
                  !  Terminal ID ..... ________              !
                  !  Set Library ..... ________              !
                  !  Set User ........ ________              !
                  !                                          !
          Code .. !  Command ===>                            !
  Object  Name .. !                                          !
                  +-----------------------------------------+ N
  Source  Library ... SYSTEM__    Database .... 10___  File .. 32___
  Target  Library ... ________    Database .... 10___  File .. 32___
  Options Replace ... N         + Criteria .... Y

Command ===>




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

Enter the required selection criteria. If one or more object-type codes have been entered in the **Object Type** field in the **Additional Criteria** window, the **Type** field in the **Programming Objects** menu is preset to the same object-type codes.

2 When you return to the **Programming Objects** menu, a plus sign (+) in front of the **Criteria** field indicates that additional object selection criteria have been specified; in the example above, user ID `SAG`.

The plus character (+) is not displayed if only the **Object Type** field contains an entry since this entry already appears in the **Type** field in the **Programming Objects** menu.

## Direct Command Syntax for Programming Objects

This section shows the syntax that applies when performing a SYSMAIN function for programming objects by using direct commands in either online or batch mode. For general instructions on using direct commands, refer to ***Executing Commands***.

For explanations of the keywords and variable values used in the syntax diagrams below, refer to ***Keywords and Variables in Direct Commands***. The symbols in the syntax diagrams correspond to the syntax symbols used for system commands. These symbols are explained in *System Command Syntax* in the *System Commands* documentation.

The syntax of the `where-clause` and the `with-clause` are identical for each command.

This section covers the following topics:

- COPY and MOVE
- DELETE
- FIND, LIST and LISTLIB
- RENAME
- where-clause
- with-clause

### COPY and MOVE

```
{ COPY }  [ { ALL       } ]  name [ FM [LIBRARY]  ]      [where-clause]
{ MOVE }  [ { CATALOGED } ]        lib-name       ]
          [ { SAVED     } ]                              lib-name [where-clause]
          [ { STOWED    } ]        TO [LIBRARY]          [with-clause]
```

**Examples:**

```
COPY PROG1 FM TESTORD TO ORDERS DBID 1 FNR 6 REP
```

```
C PGM* WITH REP TYPE PNS FM PRODLIB TO TESTLIB
```

```
M PROG1 FM OLDLIB TO NEWLIB
```

```
MOVE STOWED * TO NEWLIB WHERE DBID 100 FNR 160 FMDATE 2007-01-01 FM OLDLIB WITH ↩
XREF Y
```

## DELETE

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

**Examples:**

```
DELETE C M> IN LIB ORDERS
```

```
D * IN TESTLIB DBID 1 FNR 5 NAME SYSNAT
```

```
D SA * IN LIBTEST TYPE GLA
```

```
D * TYPE PM IN TESTORD FMDATE 2007-01-01 TODATE 2007-04-30
```

## FIND, LIST and LISTLIB

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

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

**Examples:**

```
FIND SAVED MENU IN TESTLIB
```

```
FIND STOWED MAINMENU IN SYS* WHERE DBID 1 FNR 5
```

```
F ALL PROG2 IN PROD* FNR 27 DBID 1
```

```
LIST * IN TESTLIB
```

```
LIST DT* IN TESTLIB
```

```
L SAVED TEST* IN TESTLIB TYPE PNS FNR 6
```

```
L SA TEST* TYPE PM IN TESTLIB FNR 6 DBID 2 FMDATE 2007-01-01
```

```
LISTLIB ALL MENU IN SYS* DBID 10 FNR 44
```

## RENAME

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

**Examples:**

```
RENAME PGM1 AS PROG1 FM TESTLIB
```

```
R PGM1 AS PROG1 FM TESTLIB DBID 1 FNR 5 TO PRODLIB DBID 2 FNR 6
```

```
R PGM* TYPE PS RCOP FM TESTLIB TO PRODLIB
```

### where-clause

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

**Separators**

Commas must be used as separators between the values following the `DIC` and `SEC` keywords, or if a value is missing. For example: `DIC (10,,secret,2a)`.

If the session parameter `ID` (see *ID - Input Delimiter Character* in the *Parameter Reference* documentation) has been set to a comma, use a slash (/) as the separator between values.

### with-clause

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

# 87 Processing Debug Environments


- Fields in the Debug Environments Menu ..... 588
- Selection Lists for Debug Environments ..... 590
- Direct Command Syntax for Debug Environments ..... 591

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

The debug environment specification must always correspond to the database ID (DBID) and file number (FNR) of the relevant FUSER system file.

This section describes menu functions and selection list options provided to perform a SYSMAIN function on debug environments and the syntax that applies when using direct commands.

**Related Topic:**

*Debug Environment Maintenance - Debugger* documentation

## Fields in the Debug Environments Menu

The **Debug Environments** menu contains all SYSMAIN functions required for the processing of debug environments. The fields provided in the menu are described in the following table:

| Field | Explanation | |
|---|---|---|
| **Code** | Specifies the function to be performed as described in ***Description of Functions***: | |
| | | |
| | C | Copy debug environment. |
| | D | Delete debug environment. |
| | L | List debug environment. |
| | M | Move debug environment. |
| | R | Rename debug environment. |
| | | |
| | **Note:** When a debug environment has been moved or copied from one library to another, the breakpoints and watchpoints must be adapted to the new library. For details, refer to *Maintaining Debug Environments in Different Libraries* in the *Debugger* documentation. | |
| **Sel. List** | Specifies whether selective processing or automated processing is used: | |
| | | |
| | Y | Yes.<br>Selective processing is activated. A selection list is displayed when processing debug environments. This is the default setting. For the columns, fields and line commands available on a selection list, see ***Selection Lists for Debug Environments***. |
| | N | No.<br>Selective processing is deactivated. |

| Field | Explanation |
|---|---|
| **Name** | The name of the debug environment to be processed or a range of names: see also *Specifying a Range of Names*.<br><br>The default setting is an asterisk (*) which means that all names are selected. |
| **New Name** | The name to be given to a debug environment when it is renamed with the rename function. |
| **Library** | The name of a source or a target library.<br><br>The source library contains the debug environment to be processed. The target library is the library to which the debug environment is to be copied or moved, or where the debug environment is renamed.<br><br>See also *Listing and Selecting Libraries*. |
| **Database** | The database ID (DBID) of a source or a target database.<br><br>The source database contains the library and system file where the debug environment to be processed is stored. The target database contains the library and system file to which the debug environment is to be copied or moved, or where the debug environment is renamed.<br><br>Valid database IDs are `1` to `65535`. |
| **File** | The file number (FNR) of a source or a target system file (FNAT or FUSER).<br><br>Valid file numbers are `1` to `65535`.<br><br>The source file contains the library where the debug environment to be processed is stored. The target file contains the library to which the debug environment is to be copied or moved, or where the debug environment is renamed. |
| **Replace** | Specifies whether a debug environment is to be replaced when using the move, copy or rename function:<br><br>`Y` — Yes. A debug environment with the same name which exists in the target environment is replaced.<br><br>`N` — No. A debug environment with the same name which exists in the target environment is not replaced. This is the default setting.<br><br>See also *Using the Replace Option*. |

# Selection Lists for Debug Environments

If selective processing has been activated, a selection list of all debug environments that meet the specified selection criteria is displayed on a **Selection** screen.

This section describes the columns and fields contained on a **Selection** screen and the line commands provided to further process a debug environment:

- Columns and Fields
- Line Commands

## Columns and Fields

The following columns and fields are displayed on a **Selection** screen:

| Column/Field | Explanation |
|---|---|
| **C** | Input field for line commands (see below). |
| **Environment** | The name of the debug environment that meets the specified selection criteria. |
| **Message** | The message that indicates the processing status of a debug environment. For possible messages, see ***Status Messages***. |
| **Listed Library** | The name of the library that contains the selected debug environment(s). |

## Line Commands

One of the following line commands can be entered in the **C** (Command) column of a **Selection** screen:

| Line Command | Function |
|---|---|
| A | Process the debug environment.<br><br>This line command is not available on the **List Selection** screen. On a **List Selection** screen, you can only enter a period (.) to leave the screen. |
| L | Display a list of the debug environments contained in the specified library.<br><br>Only applies to a **Library Selection** screen (see also ***Listing and Selecting Libraries***). |

# Direct Command Syntax for Debug Environments

This section shows the syntax that applies when performing a SYSMAIN function on debug environments by using direct commands in either online or batch mode. For general instructions on using direct commands, refer to ***Executing Commands***.

For explanations of the keywords and variable values used in the syntax diagrams below, refer to ***Keywords and Variables in Direct Commands***. The symbols in the syntax diagrams correspond to the syntax symbols used for system commands. These symbols are explained in *System Command Syntax* in the *System Commands* documentation.

The syntax of the `where-clause` and the `with-clause` are identical for each command.

This section covers the following topics:

- COPY and MOVE
- DELETE
- LIST
- RENAME
- where-clause
- with-clause

## COPY and MOVE

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

**Examples:**

```
COPY D ENV FM TESTLIB WHERE DBID 1 FNR 5 TO PRODLIB WHERE DBID 2 FNR 5 WITH REP
```

```
C DEBUG ENV FM TESTLIB FNR 6 TO PRODLIB FNR 7 REP
```

```
MOVE DEBUG ENV FM OLDLIB WHERE DBID 1 FNR 5 TO NEWLIB WHERE DBID 2 FNR 5
```

```
M DEBUG ENV FM OLDLIB FNR 6 TO NEWLIB FNR 7 REP
```

## DELETE

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

**Examples:**

```
DELETE DEBUG U* IN TESTLIB FNR 150
```

```
D DEBUG TEST* IN TESTLIB IN DBID 177 FNR 205
```

## LIST

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

**Examples:**

```
LIST DEBUG ENV* IN TESTLIB DBID 1 FNR 5
```

```
L D DT* IN TESTLIB DBID 10
```

## RENAME

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

**Examples:**

```
RENAME D OLDENV AS NEWENV IN TESTLIB RCOP
```

```
R DEBUG OLDENV AS NEWENV IN TESTLIB DBID 1 FNR 4 TO PRODLIB DBID 1 FNR 5
```

```
R DEBUG OLDENV AS NEWENV IN TESTLIB FNR 4 TO PRODLIB FNR 5 REPLACE RCOP
```

### where-clause

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

### with-clause

```
[WITH] [REPLACE] [RCOP] [ PROMPT   ] [ MON   ] [HELP]
                        [ NOPROMPT ] [ NOMON ]
```

# 88 Processing Error Messages


- Fields in Error Message Menus ..... 596
- Selection Lists for Error Messages ..... 598
- Renumbering Error Messages ..... 599
- Specifying Languages ..... 600
- Direct Command Syntax for Error Messages ..... 601

All SYSMAIN functions can be performed on user-defined and Natural system error messages. Error messages can be moved from one library to another, or the language texts of each error message can be copied, moved or replaced. In addition, it is possible to renumber a single error message or a range of error messages by using the rename function.

Only authorized users can process Natural system error messages if Natural Security is installed.

Error messages are stored in the system files according to their type: Natural system error messages are stored in the FNAT system file and user-defined error messages in the FUSER or FNAT system file.

This section describes menu functions and selection list options provided to perform a SYSMAIN function on error messages and the syntax that applies when using direct commands.

**Related Topic:**

*SYSERR Utility* documentation.

## Fields in Error Message Menus

The **Error Message Texts** menus contain all SYSMAIN functions required for the processing of error messages. The fields that can be provided in a menu are described in the following table:

| Field | Explanation | |
|---|---|---|
| **Code** | Specifies whether a short and/or an extended (long) error message is to be processed. Possible values are: | |
| | | |
| | A | Any error message for which a short and/or an extended (long) message exists is processed. |
| | E | Any error message for which an extended (long) error message exists is processed. Only extended error messages with corresponding short error messages are processed.<br><br>An extended error message cannot be transferred to a target environment if there is no corresponding short error message in the target environment. |
| | S | Any error message for which a short error message exists is processed. |
| **Sel. List** | Specifies whether selective processing or automated processing is used: | |
| | | |
| | Y | Yes.<br>Selective processing is activated. A selection list is displayed when processing error messages. This is the default setting. For the |

| Field | Explanation | |
|---|---|---|
| | | columns and line commands available on a selection list, see *Selection Lists for Error Messages*. |
| | N | No.<br>Selective processing is deactivated. |
| **No. From** or **Number** | The number of the error message to be processed, or the start number of a range of numbers if an end number is entered in the **No. To** field. See also *Renumbering Error Messages*. | |
| **No. To** | The end number of a range of error messages numbers to be processed if a start number is entered in the **No. From** field. See also *Renumbering Error Messages*. | |
| **New From** | Applies to the rename function.<br><br>The new number to be given to an existing error message, or the start number of a range of new numbers to be given to a range of existing error messages. See also *Renumbering Error Messages*. | |
| **New To** | Applies to the rename function.<br><br>The end number of a range of new error message numbers to be given to a range of existing error message. See also *Renumbering Error Messages*. | |
| **Library** | The name of a source or a target library or a range of names. If you want to process Natural system error messages, leave the **Library** field empty.<br><br>The source library contains the error message(s) to be processed. The target library is the library to which the error message is to be copied or moved, or where the error message is renumbered.<br><br>See also *Listing and Selecting Libraries*. | |
| **Lang.** | The language code(s) in which the error message is available.<br><br>The languages can be specified using any combination of language codes. For information on which language code is assigned to which language, see *Language Code Assignments* in **LANGUAGE* in the *System Variables* documentation. For Natural system extended (long) error messages, only language code 1 is available. Enter an asterisk (*) to select error messages in all existing languages.<br><br>See also *Specifying Languages*. | |
| **Database** | The database ID (DBID) of a source or a target database.<br><br>The source database contains the library and system file where the error message to be processed is stored. The target database contains the library and system file to which the error message is to be copied or moved, or where the error message is renumbered.<br><br>Valid database IDs are 1 to 65535. | |
| **File** | The file number (FNR) of a source or a target system file.<br><br>Valid file numbers are 1 to 65535. | |

| Field | Explanation |
|---|---|
| | The source file contains the library where the error message to be processed is stored. The target file contains the library to which the error message is to be copied or moved, or where the error message is renumbered. |
| **Replace** | Specifies whether an error message is to be replaced when using the move, copy or rename function:<br>Y — Yes. An error message with the same number which exists in the target environment is replaced.<br>N — No. An error message with the same number which exists in the target environment is not replaced. This is the default setting.<br>See also *Using the Replace Option*. |

## Selection Lists for Error Messages

If selective processing has been activated, a selection list of all error messages that meet the specified selection criteria is displayed on a **Selection** screen.

This section describes the columns contained on a **Selection** screen and the line commands provided to further process an error message:

- Columns
- Line Commands

### Columns

The following columns are displayed on a **Selection** screen:

| Column | Explanation |
|---|---|
| **C** | Input field for line commands (see below). |
| **Error** | The name of the error message that meets the specified selection criteria. |
| **Error Message Text** | The text of the short error message. |
| **Type** | The type of error message: S (short) and/or E (extended/long). |
| **Lang** | The first language code in sequence or alphabetical order that exists for the error message. If you want to display all languages that exist for an error message, on a **Selection** screen, enter the line command L or use the list function. |

| Column | Explanation |
|---|---|
| **Message** | The message that indicates the processing status of an error message.<br><br>This column only appears after a line command was executed on an error message.<br><br>For possible messages, see *Status Messages*. |

### Line Commands

One of the following line commands can be entered in the **C** (Command) column of a **Selection** screen:

| Line Command | Function |
|---|---|
| A | Process any type of message listed in the **Type** column. The type can be: short (S), extended/long (E) or short and extended (S/E). |
| E | Process the extended (long) message if the corresponding short message exists in the target environment.<br><br>If E is specified for a message which exists only as a short message, an error is returned. |
| S | Process all short messages. |
| L | Review an error message before processing it. The short and/or extended (long) error message is displayed for all existing languages, depending on the function specified previously in the **Error Message Texts** menu.<br><br>This line command can also be used on a **Library Selection** screen (see also *Listing and Selecting Libraries*). |

## Renumbering Error Messages

You can renumber a single error message or a range of error messages.

#### To renumber a single error message

1. In the **No. From** field, enter the number of the error message you want to renumber and, in the **New From** field, enter the new error message number.
2. In the **Source Library** field, enter the name of the library that contains the error message to be renumbered.

   If you want the renumbered error message to be placed in a different library, enter a name in the **Target Library** field.

### To renumber a range of error messages

1 In the **No. From** field, enter the start number of the range of error messages to be renumbered and, in the **No. To** field, enter the end number.

   In the **New From** field, enter the start number of a new range of error messages, and in the **New To** field, enter the end number.

   The number of error messages specified by the range in the **No. From** and **No. To** fields of the source library must be equal in number to the range in the **New From** and **New To** fields of the target library. For example, it is not possible to renumber error message numbers 1 - 6 as new error message numbers 7 - 10.

   If you want to renumber a range of error messages within a single library, range values must not overlap. For example, it is not possible to renumber error message numbers 1 - 6 as new message numbers 5 - 10.

2 In the **Source Library** field, enter the name of the library that contains the error messages to be renumbered.

   If you want the renumbered error messages to be placed in a different library, enter a name in the **Target Library** field.

See also the examples of range specifications in ***RENAME*** in *Direct Command Syntax for Error Messages*.

## Specifying Languages

When specifying languages consider the following:

- If the list function is performed and the specified language code does not exist for the error message, SYSMAIN uses the default language specified with the system variable `*LANGUAGE` (see **LANGUAGE* in the *System Variables* documentation).
- If an asterisk (*) is specified for a source error message, all language codes defined for this error message will overwrite any language codes defined for the target error message.

  For example: If the source error message exists only in languages `1`, `2` and `3`, and for the target error message only languages `1`, `4` and `6` are defined, after performing a copy function, the resulting target error message exists only in languages `1`, `2` and `3`.
- If the languages are specified as individual codes, each occurrence of language code is processed individually.

  For example: If the source error message contains languages `1`, `2` and `3`, and the language codes are set to `123`, and if the target error message contains languages `1`, `4` and `6`, and the language

codes are also set to 123, after performing a copy function, the resulting target error message contains languages 1, 2, 3, 4 and 6, but only the English target error message (language 1) is overwritten by the English text of the source error message.

- If a single language code is specified for the source error message and multiple language codes are specified for the target error message, after performing the copy function, the resulting target error message is in the first language specified for the target.

## Direct Command Syntax for Error Messages

This section shows the syntax that applies when performing a SYSMAIN function on error messages by using direct commands in either online or batch mode. For general instructions on using direct commands, refer to ***Executing Commands***.

For explanations of the keywords and variable values used in the syntax diagrams below, refer to ***Keywords and Variables in Direct Commands***. The symbols in the syntax diagrams correspond to the syntax symbols used for system commands. These symbols are explained in *System Command Syntax* in the *System Commands* documentation.

The syntax of the *where-clause* and the *with-clause* are identical for each command.

**Important:** For system error messages, specify NATURAL-SYSTEM or NATURAL-SYS as *lib-name*.

This section covers the following topics:

- COPY and MOVE
- DELETE
- FIND
- LIST
- RENAME
- where-clause
- with-clause

### COPY and MOVE

```
{ COPY } ERROR  number  [THRU number]
{ MOVE }                FM [LIBRARY] lib-name [where-clause]
                        TO [LIBRARY] lib-name [where-clause] [with-clause]
```

**Examples:**

```
COPY ERROR 1 FM ACCOUNTS TO ACCOUNTS1 REP WITH TYPE A
```

```
C ERROR 1 THRU 50 FM ACCT WHERE DBID 1 FNR 10 LANG 123456 TO ACCT WHERE DBID 5 FNR ↩
26 LANG 234567 WITH REP HELP
```

```
MOVE E 200 THRU 210 FM ACCT FNR 10 LANG 123 TO ACCT LANG 123 TYPE S
```

```
M E 376 TYPE E FM ACCT LANG E TO ACCT LANG G
```

## DELETE

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

**Examples:**

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

```
D E 100 IN ACCT
```

## FIND

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

**Examples:**

```
FIND E 4280 IN A* MON
```

```
F ERROR 10 IN LIB ACCT WHERE DBID 1 FNR 3 WITH TYPE E
```

## LIST

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

**Examples:**

```
LIST E 1 THRU 10 IN ACCT
```

```
L ERROR 100 THRU 150 IN LIB ACCT WHERE DBID 12 FNR 5
```

## RENAME

When renumbering a range of error messages within a single (source) library, the range values must not overlap as demonstrated in ***Examples of Invalid Number Ranges***.

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

**Examples:**

```
RENAME ERR 1 AS 101 IN ACCT
```

```
R ERROR 1 THRU 100 AS 101 THRU 200 IN CLAIMS
```

```
R ERROR 101 THRU 200 AS 1 THRU 100 IN CLAIMS
```

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

**Examples of Invalid Number Ranges:**

The following examples are *invalid*, because the number ranges overlap:

```
R ERROR 1 THRU 100 AS 51 THRU 150 IN CLAIMS
```

```
R ERROR 101 THRU 200 AS 51 THRU 150 IN CLAIMS
```

### where-clause

```
[WHERE] [DBID dbid] [FNR fnr] [NAME vsam-name] [CIPHER cipher]
[ { PASSWORD } password ] [LANGUAGE language] [SEC (dbid,fnr,password,cipher)]
  { PSW      }                                
```

**Separators**

Commas must be used as separators between the values following the `SEC` keyword, or if a value is missing. For example: `SEC (10,,secret,2a)`. If the session parameter `ID` (see *ID - Input Delimiter Character* in the *Parameter Reference* documentation) has been set to a comma, use a slash (/) as the separator between values.

### with-clause

```
[WITH] [TYPE type] [REPLACE] [ RCOP] [ PROMPT ] [ MON ] [HELP]
                                [ NOPROMPT ] [ NOMON ]
```

# 89 Processing Profiles


- Fields in the Profiles Menu ..... 606
- Selection Lists for Profiles ..... 608
- Direct Command Syntax for Profiles ..... 609

All SYSMAIN functions except the find function can be performed on the following types of profile: device profiles, editor profiles, map profiles and parameter profiles. Device, editor and map profiles are created with the program editor or the map editor. Parameter profiles are created with the SYSPARM utility.

This section describes menu functions and selection list options provided to perform a SYSMAIN function on profiles and the syntax that applies when using direct commands.

**Related Topics:**

*Editors* documentation and ***SYSPARM Utility*** documentation

## Fields in the Profiles Menu

The **Profiles** menu contains all SYSMAIN functions required for the processing of profiles. The fields provided in the menu are described in the following table:

| Field | Explanation |
|---|---|
| **Code** | Specifies the function to be performed as described in ***Description of Functions***:<br>C — Copy profile.<br>D — Delete profile.<br>L — List profile.<br>M — Move profile.<br>R — Rename profile. |
| **Sel. List** | Specifies whether selective processing or automated processing is used:<br>Y — Yes. Selective processing is activated. A selection list is displayed when processing profiles. This is the default setting. For the columns and line commands available on a selection list, see ***Selection Lists for Profiles***.<br>N — No. Selective processing is deactivated. |
| **Name** | The name of the profile to be processed or a range of names: see also ***Specifying a Range of Names***. |
| **New Name** | The name to be given to a profile when it is renamed with the rename function. |
| **Type** | The type of profile to be processed:<br>D — Only device profiles are processed. |

| Field | Explanation |
|---|---|
| | E — Only editor profiles are processed.<br>M — Only map profiles are processed.<br>P — Only parameter profiles are processed.<br>* or empty field — All profiles are processed. This is the default setting.<br><br>Types D, E, M and P can be used in any combination. |
| **Database** | The database ID (DBID) of a source or a target database.<br><br>The source database contains the system file where the profile to be processed is stored. The target database contains the system file to which the profile is to be copied or moved, or where the profile is renamed.<br><br>Valid database IDs are 1 to 65535. |
| **File** | The file number (FNR) of a source or a target system file.<br><br>Valid file numbers are 1 to 65535.<br><br>The source file contains the profile to be processed. The target file is the file to which the profile is to be copied or moved, or where the profile is renamed. |
| **Name** (below **New Name**) | Only applies to VSAM files.<br><br>The DDNAME/FCT entry for the source or target file number. |
| **Replace** | Specifies whether a profile is to be replaced when using the move, copy or rename function:<br><br>Y — Yes. A profile with the same name which exists in the target environment is replaced.<br>N — No. A profile with the same name which exists in the target environment is not replaced. This is the default setting.<br><br>See also *Using the Replace Option*. |

# Selection Lists for Profiles

If selective processing has been activated, a selection list of all profiles that meet the specified selection criteria is displayed on a **Selection** screen.

This section describes the columns contained on a **Selection** screen and the line commands provided to further process a profile:

- Columns
- Line Commands

## Columns

The following columns are displayed on a **Selection** screen:

| Column | Explanation |
|---|---|
| **C** | Input field for line commands (see below). |
| **Profile** | The name of the profile that meets the specified selection criteria. |
| **Type** | The type of profile:<br>D — Device profile.<br>E — Editor profile.<br>M — Map profile.<br>P — Parameter profile. |
| **Message** | The message which indicates the current processing status of a profile. For possible messages, see *Status Messages*. |

## Line Commands

The line commands that can be entered in the **C** (Command) column of a **Selection** screen are listed in the following table. Line commands `E`, `M`, `D` and `P` can be entered in any combination.

| Line Command | Function |
|---|---|
| `A` | Process device profiles, editor profiles, map profiles and/or parameter profiles. |
| `D` | Process only device profiles, even if there are editor, map and parameter profiles listed as well. If `D` is specified for a profile which does not exist as a device profile, an error occurs. |
| `E` | Process only editor profiles, even if there are device, map and parameter profiles listed as well. If `E` is specified for a profile which does not exist as an editor profile, an error is returned. |

| Line Command | Function |
|---|---|
| M | Process only map profiles, even if there are device, editor and parameter profiles listed as well. If M is specified for a profile which does not exist as a map profile, an error is returned. |
| L | Display the contents of a parameter profile. |
| P | Process only parameter profiles, even if there are device, editor and map profiles listed as well. If P is specified for a profile which does not exist as a parameter profile, an error is returned. |

## Direct Command Syntax for Profiles

This section shows the syntax that applies when performing a SYSMAIN function on profiles by using direct commands in either online or batch mode. For general instructions on using direct commands, refer to ***Executing Commands***.

For explanations of the keywords and variable values used in the syntax diagrams below, refer to ***Keywords and Variables in Direct Commands***. The symbols in the syntax diagrams correspond to the syntax symbols used for system commands. These symbols are explained in *System Command Syntax* in the *System Commands* documentation.

The syntax of the `where-clause` and the `with-clause` are identical for each command.

This section covers the following topics:

- COPY and MOVE
- DELETE
- LIST
- RENAME
- where-clause
- with-clause

### COPY and MOVE

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

**Examples:**

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

## DELETE

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

**Examples:**

```
DELETE PRO U* TYPE DM
D PRO TEST* IN DBID 177 FNR 205
```

## LIST

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

**Examples:**

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

## RENAME

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

**Examples:**

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

### where-clause

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

### with-clause

```
[WITH] [TYPE type] [REPLACE] [RCOP] [ PROMPT   ] [ MON   ] [HELP]
                                    [ NOPROMPT ] [ NOMON ]
```

# 90 Processing Rules


- Fields in the Rules Menu ..... 614
- Selection Lists for Rules ..... 615
- Direct Command Syntax for Rules ..... 617

All SYSMAIN functions except the find function can be performed on automatic and free rules.

If Predict is installed, we recommend that you use Predict instead of SYSMAIN for the processing of rules; Natural does not process all information on rules (for example, format and verification type) provided by Predict.

The rule environment specification must always correspond to the database ID (DBID) and file number (FNR) of the relevant FDIC system file.

This section describes menu functions and selection list options provided to perform a SYSMAIN function on rules and the syntax that applies when using direct commands.

## Fields in the Rules Menu

The **Rules** menu contains all SYSMAIN functions required for the processing of rules. The fields provided in the menu are described in the following table:

| Field | Explanation |
|---|---|
| **Code** | Specifies the function to be performed as described in *Description of Functions*:<br>C — Copy rule.<br>D — Delete rule.<br>L — List rule.<br>M — Move rule.<br>R — Rename rule. |
| **Sel. List** | Specifies whether selective processing or automated processing is used:<br>Y — Yes. Selective processing is activated. A selection list is displayed when processing rules. This is the default setting. For the columns and line commands available on a selection list, see *Selection Lists for Rules*.<br>N — No. Selective processing is deactivated. |
| **Name** | The name of the rule to be processed or a range of names: see also *Specifying a Range of Names*. |
| **New Name** | The name to be given to a rule when it is renamed with the rename function. |
| **Type** | The type of rule to be processed:<br>A — Only automatic rules are processed. |

| Field | Explanation | |
|---|---|---|
| | F | Only free rules are processed. |
| | AF<br>*<br>or<br>empty field | All rules are processed: automatic and/or free rules. This is the default setting. |
| **Database** | The database ID (DBID) of a source or a target database.<br><br>The source database contains the FDIC system file where the rule to be processed is stored. The target database contains the FDIC system file to which the rule is to be copied or moved, or where the rule is renamed.<br><br>Valid database IDs are 1 to 65535. | |
| **File** | The file number (FNR) of a source or a target FDIC system file.<br><br>Valid file numbers are 1 to 65535.<br><br>The source file contains the rule to be processed. The target file is the file to which the rule is to be copied or moved, or where the rule is renamed. | |
| **Name**<br>(next to **File**) | Only applies to VSAM files.<br><br>The DDNAME/FCT entry for the source or target file number. | |
| **Replace** | Specifies whether a rule is to be replaced when using the move, copy or rename function: | |
| | | |
| | Y | Yes.<br>A rule with the same name which exists in the target environment is replaced. |
| | N | No.<br>A rule with the same name which exists in the target environment is not replaced. This is the default setting. |
| | | |
| | See also *Using the Replace Option*. | |

## Selection Lists for Rules

If selective processing has been activated, a selection list of all rules that meet the specified selection criteria is displayed on a **Selection** screen.

This section describes the columns contained on a **Selection** screen and the line commands provided to further process a rule:

- Columns

- Line Commands

## Columns

The following columns are displayed on a **Selection** screen:

| Column | Explanation |
|---|---|
| **C** | Input field for line commands (see below). |
| **Rule Name** | The name of the rule that meets the specified selection criteria. |
| **Type** | The type of rule: `Free` or `Auto` (Automatic) rule. |
| **Ver. Type** | The verification type:<br>an attribute of the rule which is directly taken from the verification as defined in Predict. A value of `Unknown` indicates that there is no corresponding verification information in Predict available. For valid values, see the *Predict* documentation. |
| **Format** | The format type:<br>an attribute of the rule which is directly taken from the verification as defined in Predict. A value of `Unknown` indicates that there is no corresponding verification information in Predict available. For valid values, see the relevant *Predict* documentation. |
| **Message** | The message which indicates the current processing status of a rule. For possible messages, see ***Status Messages***. |

## Line Commands

One of the following line commands can be entered in the **C** (Command) column of a **Selection** screen:

| Line Command | Function |
|---|---|
| `A` | Process the specified rule. |
| `F` | Only applies to automatic rules.<br><br>All DDMs and fields which have the specified rule assigned are displayed. |
| `H` | Produce a hardcopy of the saved (source) object of a rule.<br><br>The source code of the specified rule is printed and listed on the screen. |
| `I` | Display directory information of the rule.<br><br>This command is similar to the system command `LIST DIR` (see *Displaying Directory Information* in the *System Commands* documentation). |
| `L` | Display the source code of a rule. |

## Direct Command Syntax for Rules

This section shows the syntax that applies when performing a SYSMAIN function on rules by using direct commands in either online or batch mode. For general instructions on using direct commands, refer to ***Executing Commands***.

For explanations of the keywords and variable values used in the syntax diagrams below, refer to ***Keywords and Variables in Direct Commands***. The symbols in the syntax diagrams correspond to the syntax symbols used for system commands. These symbols are explained in *System Command Syntax* in the *System Commands* documentation.

The syntax of the `where-clause` and the `with-clause` are identical for each command.

This section covers the following topics:

- COPY and MOVE
- DELETE
- LIST
- RENAME
- where-clause
- with-clause

### COPY and MOVE

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

**Examples:**

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

```
COPY R C< FM FNR 20 TO FNR 24
```

```
M RULE TESTRULE FM FNR 20 TO FNR 24 REPLACE
```

```
MOVE RULE C< FM FNR 20 TO FNR 24
```

## DELETE

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

**Example:**

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

## LIST

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

**Example:**

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

## RENAME

```
RENAME   RULE  name  AS new-name [ FM          ] [ TO          ] [with-clause]
                                 [ where-clause] [ where-clause]
```

**Examples:**

```
RENAME R OLDRULE AS NEWRULE FM DBID 1 FNR 4 TO DBID 1 FNR 5
```

```
R RULE OLDRULE AS NEWRULE FM FNR 4 TO FNR 5 REPLACE RCOP
```

## where-clause

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

### Separators

Commas must be used as separators between the values following the `DIC` and `SEC` keywords, or if a value is missing. For example: `DIC (10,,secret,2a)`. If the session parameter `ID` (see *ID - Input Delimiter Character* in the *Parameter Reference* documentation) has been set to a comma, use a slash (/) as the separator between values.

### with-clause

```
[WITH] [TYPE type] [REPLACE] [RCOP] [ PROMPT   ] [ MON   ] [HELP]
                                    [ NOPROMPT ] [ NOMON ]
```

# 91 Processing DL/I Subfiles


- Fields in the DL/I Subfiles Menu ..... 622
- Selection Lists for DL/I Subfiles ..... 623
- Direct Command Syntax for DL/I Subfiles ..... 624

All SYSMAIN functions except the find and rename functions can be performed on DL/I subfiles if Natural for DL/I is installed.

DL/I subfiles include the following:

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

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

This section describes menu functions and selection list options provided to perform a SYSMAIN function on DL/I subfiles and the syntax that applies when using direct commands.

## Fields in the DL/I Subfiles Menu

The **DL/I Subfiles** menu contains all SYSMAIN functions required for the processing of DL/I subfiles. The fields provided in the menu are described in the following table:

| Field | Explanation |
|---|---|
| **Code** | Specifies the function to be performed as described in ***Description of Functions***:<br>C — Copy subfile.<br>D — Delete subfile.<br>L — List subfile.<br>M — Move subfile. |
| **Sel. List** | Specifies whether selective processing or automated processing is used:<br>Y — Yes. Selective processing is activated. A selection list is displayed when processing DL/I subfiles. This is the default setting. For the columns, fields and line commands available on a selection list, see ***Selection Lists for DL/I Subfiles***.<br>N — No. Selective processing is deactivated. |
| **Name** | The name of the DL/I subfile to be processed or a range of names: see also ***Specifying a Range of Names***. |
| **Type** | The type of DL/I subfile to be processed. You must specify either of the following types: |

| Field | Explanation | |
|---|---|---|
| | D | Only NDBs and UDFs are processed. This is the default setting. |
| | P | Only NSBs are processed. |
| **Database** | The database ID (DBID) of a source or a target database.<br><br>The source database contains the FDIC system file where the DL/I subfile to be processed is stored. The target database contains the FDIC system file to which the DL/I subfile is to be copied or moved.<br><br>Valid database IDs are 1 to 65535. | |
| **File** | The file number (FNR) of a source or a target FDIC system file.<br><br>Valid file numbers are 1 to 65535.<br><br>The source file contains the DL/I subfile to be processed. The target file is the file to which the DL/I subfile is to be copied or moved. | |
| **Name** (next to **File**) | Only applies to VSAM files.<br><br>The DDNAME/FCT entry for the source or target file number. | |
| **Replace** | Specifies whether a DL/I subfile is to be replaced when using the move or copy function: | |
| | | |
| | Y | Yes.<br>A DL/I subfile with the same name which exists in the target environment is replaced. |
| | N | No.<br>A DL/I subfile with the same name which exists in the target environment is not replaced. This is the default setting. |
| | | |
| | See also *Using the Replace Option*. | |

## Selection Lists for DL/I Subfiles

If selective processing has been activated, a selection list of all DL/I subfiles that meet the specified selection criteria is displayed on a **Selection** screen.

This section describes the columns and fields contained on a **Selection** screen and the line commands provided to further process a DL/I subfile:

- Columns and Fields

- Line Commands

### Columns and Fields

The following columns and fields are displayed on a **Selection** screen:

| Column/Field | Explanation |
|---|---|
| **C** | Input field for line commands (see below). |
| **Subfile** | The name of the DL/I subfile that meets the specified selection criteria. |
| **Message** | The message that indicates the processing status of a DL/I subfile. For possible messages, see ***Status Messages***. |
| **Listed Subfiles** | The type of DL/I subfile: `NDB` (including UDFs) or `NSB`. |

### Line Commands

One of the following line commands can be entered in the **C** (Command) column of a **Selection** screen:

| Line Command | Function |
|---|---|
| `A` | Process NSB subfile or NDB and UDF subfiles.<br><br>Line command `A` is not available with the list function. |
| `L` | List NSB subfile or NDB and UDF subfiles. |

## Direct Command Syntax for DL/I Subfiles

This section shows the syntax that applies when performing a SYSMAIN function on a DL/I subfile by using direct commands in either online or batch mode. For general instructions on using direct commands, refer to ***Executing Commands***.

For explanations of the keywords and variable values used in the syntax diagrams below, refer to ***Keywords and Variables in Direct Commands***. The symbols in the syntax diagrams correspond to the syntax symbols used for system commands. These symbols are explained in *System Command Syntax* in the *System Commands* documentation.

The syntax of the *`where-clause`* and the *`with-clause`* are identical for each command.

This section covers the following topics:

- COPY and MOVE
- DELETE and LIST
- where-clause

- with-clause

### COPY and MOVE

```
{ COPY } DL1 name [with-clause] [ FM where-clause ] TO where-clause
{ MOVE }
```

**Examples:**

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

### DELETE and LIST

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

**Examples:**

```
DELETE DL1 S* TYPE D
D DL1 TEST* IN DBID 177 FNR 205 TYPE D
LIST DL1 SUBFILE* IN DBID 1 FNR 5 TYPE D
L DL1 SF* TYPE P
```

### where-clause

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

### with-clause

```
[WITH] [TYPE type] [REPLACE] [RCOP] [ PROMPT   ] [ MON   ] [HELP]
                                    [ NOPROMPT ] [ NOMON ]
```

# 92 Processing DDMs


- Fields in the DDMs Menu ..... 628
- Selection Lists for DDMs ..... 630
- Direct Command Syntax for DDMs ..... 631

All SYSMAIN functions except the find and rename functions can be performed on DDMs.

The DDM environment specification must always correspond to the database ID (DBID) and file number (FNR) of the relevant FDIC system file.

This section describes menu functions and selection list options provided to perform a SYSMAIN function on DDMs and the syntax that applies when using direct commands.

## Fields in the DDMs Menu

The **DDMs** menu contains all SYSMAIN functions required for the processing of DDMs. The fields provided in the menu are described in the following table:

| Field | Explanation |
|---|---|
| **Code** | Specifies the function to be performed as described in *Description of Functions*:<br>`C` — Copy DDM.<br>`D` — Delete DDM.<br>`L` — List DDM.<br>`M` — Move DDM. |
| **Sel. List** | Specifies whether selective processing or automated processing is used:<br>`Y` — Yes.<br>Selective processing is activated. A selection list is displayed when processing DDMs. This is the default setting.<br><br>For the columns and line commands available on a selection list, see *Selection Lists for DDMs*.<br>`N` — No.<br>Selective processing is deactivated. |
| **Name** | The name of the DDM to be processed or a range of names: see also *Specifying a Range of Names*. |
| **DDM DBID** | This field can be used to select only DDMs which were cataloged under the specified database ID (DBID).<br><br>Valid database IDs are `1` to `65535`.<br><br>If this field is empty or set to `0` (zero), the DBID is not used as a selection criterion. |
| **DDM FNR** | This field can be used to select only DDMs which were cataloged under the specified file number (FNR).<br><br>Valid file numbers are `1` to `65535`. |

| Field | Explanation |
|---|---|
| | If this field is empty or set to `0` (zero), the FNR is not used as a selection criterion. |
| **Database** | The database ID (DBID) of a source or a target database.<br><br>The source database contains the FDIC system file where the DDM to be processed is stored. The target database contains the FDIC system file to which the DDM is to be copied or moved.<br><br>Valid database IDs are `1` to `65535`. |
| **File** | The file number (FNR) of a source or a target FDIC system file.<br><br>Valid file numbers are `1` to `65535`.<br><br>The source file contains the DDM to be processed. The target file is the file to which the DDM is to be copied or moved. |
| **Name**<br>(next to **File**) | Only applies to VSAM files.<br><br>The DDNAME/FCT entry for the source or target file number. |
| **Replace** | Specifies whether a DDM is to be replaced when using the move or copy function:<br><br>`Y` — Yes. A DDM with the same name which exists in the target environment is replaced.<br><br>`N` — No. A DDM with the same name which exists in the target environment is not replaced. This is the default setting.<br><br>See also ***Using the Replace Option***. |
| **Del.NSC-Def.** | This field only appears in a Natural Security environment.<br><br>Indicates deletion of Natural Security definitions.<br><br>If a DDM is deleted from a source environment or moved to a new environment and different FSEC system files have been specified, you can use this field to specify whether or not to delete the DDM definition in the source FSEC.<br><br>Possible values are:<br><br>`Y` — Yes. The DDM definition in the source FSEC is deleted.<br><br>`N` — No. The DDM definition in the source FSEC is kept. This is the default setting. |

# Selection Lists for DDMs

If selective processing has been activated, a selection list of all DDMs that meet the specified selection criteria is displayed on a **Selection** screen.

This section describes the columns contained on a **Selection** screen and the line commands provided to further process a DDM:

- Columns
- Line Commands

## Columns

The following columns are displayed on a **Selection** screen:

| Column | Explanation |
|---|---|
| **C** | Input field for line commands (see below). |
| **DDM Name** | The name of the DDM that meets the specified selection criteria. |
| **DBID** | The database ID (DBID) under which the DDM was cataloged.<br><br>If no DBID or 0 (zero) is listed, the DDM was cataloged under the current DBID. |
| **FNR** | The file number (FNR) under which the DDM was cataloged. |
| **Cat Date** | The date on which the DDM was cataloged. |
| **Time** | The time when the DDM was cataloged. |
| **User ID** | The ID of the user who cataloged the DDM. |
| **Message** | The message that indicates the processing status of a DDM.<br><br>This column only appears after the line command A was executed on a DDM.<br><br>For possible messages, see ***Status Messages***. |

The columns **Cat Date**, **Time** and **User ID** are empty if the DDM was cataloged under an older version of Natural which does not supply this type of information.

### Line Commands

One of the following line commands can be entered in the **C** (Command) column of a **Selection** screen:

| Line Command | Function |
|---|---|
| A | Process the specified DDM.<br><br>This line command is not available with the list function. |
| L | List the specified DDM.<br><br>For detailed information on the field definitions displayed, see *Columns of Field Attributes* in the *SYSDDM Utility* documentation. |
| R | List all automatic rules linked to the specified DDM.<br><br>Line command R invokes the **Rule Usage for DDM/Fields** screen, which indicates whether a DDM uses a rule or not. If a rule is used, the name of the rule and the DDM field referenced are listed on the screen. |
| X | Only applies if Natural Connection and Entire Connection are installed.<br><br>Download DDM(s) to a PC. For detailed instructions, refer to ***Downloading Objects to a PC***. |

## Direct Command Syntax for DDMs

This section shows the syntax that applies when performing a SYSMAIN function on DDMs by using direct commands in either online or batch mode. For general instructions on using direct commands, refer to ***Executing Commands***.

For explanations of the keywords and variable values used in the syntax diagrams below, refer to ***Keywords and Variables in Direct Commands***. The symbols in the syntax diagrams correspond to the syntax symbols used for system commands. These symbols are explained in *System Command Syntax* in the *System Commands* documentation.

The syntax of the *where-clause* and the *with-clause* are identical for each command.

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

This section covers the following topics:

- COPY and MOVE
- DELETE
- LIST
- where-clause

- with-clause

## COPY and MOVE

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

**Examples:**

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

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

```
M DDM PERSONNEL FM FNR 20 TO FNR 24 REPLACE
```

```
MOVE DDM C< FM FNR 20 TO FNR 24
```

## DELETE

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

**Example:**

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

## LIST

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

**Example:**

```
L DDM * IN DBID 1 FNR 5
```

## where-clause

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

### Separators

Commas must be used as separators between the values following the `DIC` and `SEC` keywords, or if a value is missing. For example: `DIC (10,,secret,2a)`. If the session parameter `ID` (see *ID - Input Delimiter Character* in the *Parameter Reference* documentation) has been set to a comma, use a slash (/) as the separator between values.

### with-clause

```
[WITH] [ { DDMDBID } DDM-dbid ] [ { DDMFNR } DDM-fnr ] [REPLACE] [ PROMPT ] [ MON ] [HELP]
         { DDBID  }             { DFNR  }              [ NOPROMPT ] [ NOMON ]
```

# 93 Processing Predict Sets


- Fields in the Predict Sets Menu ..... 636
- Selection Lists for Predict Sets ..... 637
- Direct Command Syntax for Predict Sets ..... 638

This chapter describes the menus and direct commands to process Predict sets. The following topics are covered:

**Notes:**

1. All SYSMAIN functions except the `find` function can be performed on Predict sets.
2. The specification of the environment for Predict sets must always correspond with the database ID (DBID) and file number (FNR) of the relevant FDIC system file.
3. During the processing of Predict sets the Software AG Editor is used. Therefore you have to use the *Editor Buffer Pool* or to set the Natural profile parameter `EDPSIZE`.

## Fields in the Predict Sets Menu

The **Predict Sets** menu contains all SYSMAIN functions for processing Predict sets. The fields provided in the menu are described in the following table:

| Field | Explanation |
|---|---|
| **Code** | Specifies the function to be performed as described in *Description of Functions*<br><br>C Copy Predict set.<br>D Delete Predict Set.<br>L List Predict Set.<br>M Move Predict Set.<br>R Perform the rename function. |
| **Sel. List** | Specifies whether selective processing or automated processing is used:<br><br>Y Yes.<br>Selective processing is activated. A selection list is displayed when processing Predict sets. This is the default setting. For the columns and line commands available on a selection list, see *Selection Lists for Predict Sets*.<br>N No.<br>Selective processing is deactivated and automated processing becomes activated. |
| **Set No. from** | The number of the Predict set to be processed, or the start number of a range of Predict sets if an end number is entered in the **No. to** field. |
| **New No.** | The (new) number to be given to a Predict set for which the rename function is performed. |
| **Library** | The name of a source or a target library assigned to the Predict set(s). It is not possible to specify ranges. |
| **Database** | The database ID (DBID) of a source or a target database for the library assigned to the Predict set(s). |

| Field | Explanation |
|---|---|
| **File** | The file number (FNR) of a source or a target database for the library assigned to the Predict set(s). |
| **User** | The source or target user ID assigned to the Predict set(s). It is not possible to specify ranges. |
| **FDIC Database** | The database ID (DBID) of a source or a target database.<br><br>The source database contains the FDIC system file where the Predict set to be processed is stored. The target database contains the FDIC system file to which the Predict set is to be copied or moved, or where the **Rename** function is performed.<br><br>Valid database IDs range from `1` to `65535`. |
| **FDIC File** | The file number (FNR) of a source or a target FDIC system file.<br><br>The source file contains the Predict set to be processed. The target file is the file to which the Predict set is to be copied or moved or where the rename function is performed.<br><br>Valid file numbers range from 1 to 65535. |
| **Replace** | Specifies whether a Predict set is to be replaced when using the move, copy or rename function:<br><br>Y Yes. A Predict set in the target environment with the same number for the same library and user is replaced.<br>N No. No replacement. This is the default setting.<br><br>See also ***Using the Replace Option*** . |

## Selection Lists for Predict Sets

If selective processing has been activated, a selection list of all Predict sets that meet the specified selection criteria is displayed on a **Selection** screen.

This section describes the columns contained on a **Selection** screen and the line commands provided to further process a Predict set.

- Columns

- Line Commands

### Columns

The following columns are displayed on the **Selection** screen:

| Column | Explanation |
|---|---|
| **C** | Input field for line commands (see below) |
| **Set** | The number of the Predict set that matches the specified selection criteria. |
| **Criteria** | The criteria (text of the Predict command) used for the creation of the Predict set. |
| **Members** | Number of Natural object names contained in the Predict set. |
| **Date** | The date when the Predict Set was created. |
| **Time** | The time when the Predict Set was created. |

### Line Commands

One of the following line commands can be entered in the C (Command) column of a Selection screen:

| Line Command | Explanation |
|---|---|
| **A** | Process the specified Predict set. |
| **L** | Display the contents (names and types) of the specified Predict set. |

## Direct Command Syntax for Predict Sets

This section shows the syntax that applies when performing a SYSMAIN function on Predict sets by using direct commands in either online or batch mode. For general instructions on using direct commands, refer to ***Executing Commands***.

For explanations of the keywords and variable values used in the syntax diagrams below, refer to ***Keywords and Variables in Direct Commands***. The symbols in the syntax diagrams correspond to the syntax symbols used for system commands. These symbols are explained in *System Command Syntax* in the *System Commands* documentation.

The syntax of the `where-clause` and the `with-clause` are identical for each command.

This section covers the following topics:

- COPY and MOVE
- DELETE
- LIST

- RENAME
- where-clause
- with-clause

## COPY and MOVE

```
{ COPY }  SET nn        [ TOSETnn ]  [ FROM / FM / IN  [LIBRARY] lib-name ]        [SETUSER user]
{ MOVE }                                                                           [where-clause]
                                     TO  [LIBRARY] lib-name [SETUSER user] [where-clause]
          [with-clause]
```

**Examples:**

```
COPY SET 1 TOSET 99 FROM LIBRARY SRCLIB SETUSER UID1 WHERE DBID 10 FNR 32 DIC (10/460)
  TO LIBRARY TGTLIB SETUSER UID2 WHERE DBID 10 FNR 110 DIC (10/1460) WITH REPLACE
COPY SET 1 TOS 99 FM SRCLIB SETU UID1 TO TGTLIB SETU UID2 REPLACE
```

## DELETE

```
DELETE  SET nn        [ TOSETnn ] [ FROM / FM / IN  [LIBRARY] lib-name ] [SETUSER set-user]  [where-clause]
                [with-clause]
```

**Example:**

```
D SET 17 IN SRCLIB DBID 10 FNR 32
```

## LIST

```
LIST   SET nn [ TOSETnn ]  [ FROM / FM / IN  [LIBRARY] lib-name ] [SETUSER user] [where-clause]
              [with-clause]
```

**Example:**

```
LIST SET 11 TOS 44 IN LIB SRCLIB DBID 10 FNR 32
```

## RENAME

```
RENAME   SET nn AS nn [ FROM
                      FM    [LIBRARY] lib-name ]          [SETUSER user]
                      IN                                  [where-clause]
                      [TO   [LIBRARY] lib-name [SETUSER user] [where-clause]]
         [with-clause]
```

**Example:**

```
RENAME SET 17 AS 83 FM SRCLIB SETUSER UID1 WHERE DBID 10 FNR 32 DIC (10/1460) TO ↩
TGTLIB  ↩
                 SETUSER UID1 WHERE DBID 10 FNR 32 DIC (10/1460) WITH RCOP REP
RENAME SET 11 AS 11 FM LIB1 SETUSER UID1 TO LIB1 SETUSER UID2 WITH RCOP'
```

### where-clause

```
[WHERE] [DBID dbid] [FNR fnr]
   [ { PASSWORD } password ]
     { PSW      }
   [DIC (dbid,fnr,password,cipher)]
   [SEC (dbid,fnr,password,cipher)]
```

**Separators**

Commas must be used as separators between the values following the `DIC` and `SEC` keywords, or if a value is missing. For example: `DIC (10,,secret,2a)`. If the session parameter `ID` (see *ID - Input Delimiter Character* in the *Parameter Reference* documentation) has been set to a comma, use a slash (/) as the separator between values.

### with-clause

```
[WITH]  [REPLACE] [ PROMPT   ] [ MON   ] [HELP]
                  [ NOPROMPT ] [ NOMON ]
```

# 94 Keywords and Variables in Direct Commands


- Description of Keywords ..... 642
- Specifying a Range of Names ..... 653

This section describes all keywords and variables that are relevant when using direct commands in online or batch mode. Each keyword represents a parameter that is used to specify object selection criteria or set an option for the command being executed. If indicated, a variable value must be supplied with a keyword.

The symbols used in the syntax diagrams shown below correspond to the syntax symbols used for system commands which are explained in *System Command Syntax* in the *System Commands* documentation.

For the direct command syntax to which the keywords refer, including details on the `where-clause` and the `with-clause` mentioned in this section, refer to the object-type specific sections of the SYSMAIN Utility documentation.

## Description of Keywords

This section explains the keywords and corresponding variable values (if required) used in a direct command.

Keywords are listed alphabetically. Letters in italics represent variable values that must be supplied with a keyword. For each variable value, the Natural data format and length is indicated.

| Keyword | Value | Natural Data Format/Length | Explanation |
|---|---|---|---|
| `ALL`<br>or<br>`A` | *`name`* | A9 | Only applies to programming objects.<br><br>The name of the object to be processed or a range of names (see also ***Specifying a Range of Names***). Any saved (source) objects and/or cataloged objects are processed. |
| `AS` | *`new-name`* | A8<br>or<br>A12 | Not applicable to DL/I subfiles and DDMs.<br><br>The new name to be given to an object when it is renamed with the `RENAME` command.<br><br>Format/length A12 only applies to debug environments. |
| | *`new-number`* | N4 | For error messages:<br><br>The new number to be given to an error message, or the start number of a range of new numbers to be given to a range of existing error messages when using the `RENAME` command. |
| | *`new-set-number`* | N2 | For Predict sets:<br><br>The new number to be given to a Predict Set when using the `RENAME` command. |

| Keyword | Value | Natural Data Format/Length | Explanation |
|---|---|---|---|
| CATALOGED | *name* | A9 | Only applies to programming objects.<br><br>The name of the cataloged object to be processed or a range of names (see also ***Specifying a Range of Names***). |
| CIPHER | *cipher* | A8 | The Adabas cipher code of a source file and/or target system file which is used in a *where-clause*.<br><br>For rules and DDMs: The corresponding DIC specification can be used instead of CIPHER. If *cipher* is specified twice, the one specified last will be used. |
| DBID | *dbid* | N5 | The database ID (DBID) of a source or a target database.<br><br>The source database contains the system file where the object to be processed is stored. The target database contains the system file to which the object is to be copied or moved, or where the object is renamed (or in the case of an error message, renumbered) if relevant.<br><br>Valid DBIDs are 1 to 65535.<br><br>If no DBID or file number (**FNR**) is specified and SYSMAIN is called with the system command SYSMAIN or the subprogram MAINUSER (see also ***Invoking SYSMAIN with Appl. Programming Interface***), the following applies: The DBID and FNR of the system file from which SYSMAIN was called are always used. For example: if you enter SYSMAIN from a library contained in the FUSER system file, the DBID and FNR of this file are used.<br><br>For rules and DDMs: The corresponding DIC specification can be used instead of DBID. If *dbid* is specified twice, the one specified last will be used. |
| DDM<br>or<br>VIEW | *name* | A32 | Only applies to DDMs.<br><br>The name of the DDM to be processed or a range of names. See also ***Specifying a Range of Names***. |
| DDMDBID<br>or<br>DDBID | *ddm-dbid* | N5 | Only applies to DDMs.<br><br>The DDM database ID (DBID): All DDMs that were cataloged under the specified DBID are processed.<br><br>Valid DDM DBIDs are 0 to 65535. If no value or 0 (zero) is specified, the DDM DBID is not checked. |
| DDMFNR<br>or<br>DFNR | *ddm-fnr* | N5 | Only applies to DDMs.<br><br>The DDM file number (FNR): All DDMs that were cataloged under the specified FNR are processed. |

| Keyword | Value | Natural Data Format/Length | Explanation |
|---|---|---|---|
| | | | Valid DDM FNRs are 0 to 65535. If no value or 0 (zero) is specified, the DDM FNR is not checked. |
| DEBUG | *name* | A12 | Only applies to debug environments.<br><br>The name of the debug environment to be processed or a range of names. See also ***Specifying a Range of Names***. |
| DIC | *dbid*<br>*fnr*<br>*password*<br>*cipher* | A80 | Not applicable to error messages, profiles and DL/I subfiles.<br><br>Specifies the environment of the FDIC source and/or target system file: database ID (*dbid*), file number (*fnr*), Adabas password (*password*) and Adabas cipher code (*cipher*).<br><br>For rules and DDMs: DBID, FNR, CIPHER and PASSWORD specifications can be used instead of the corresponding DIC specifications, or vice versa. If an item is specified twice, the one specified last will be used. |
| DL1<br>or<br>SUBFILES<br>or<br>S | *name* | A8 | Only applies to DL/I subfiles.<br><br>The name of the DL/I subfile to be processed or a range of names. See also ***Specifying a Range of Names***. |
| ERROR | *number* | N4 | Only applies to error messages.<br><br>The number of the error message to be processed or the start number of a range of numbers if THRU is specified. |
| EXTEND | - | - | Only applies to programming objects and the LIST or FIND command when being used in batch mode.<br><br>If EXTEND is *not* specified, a short list of the objects contained in the specified library is displayed. The short list contains the name and the type of the object and indicates whether a source object and/or a cataloged object exists.<br><br>If EXTEND is specified, an extended list of the objects contained in the specified library is displayed. In addition to the information displayed if EXTEND is not specified, the extended list provides information from the object directory: programming mode, Natural version, user ID, saved/cataloged date and time, and the source of the object (if any). |
| FROM<br>or<br>FM<br>or<br>IN | *lib-name* | A8 | For programming objects, Predict sets, debug environments and error messages: Specifies a source library.<br><br>For profiles, rules, DDMs and DL/I subfiles: Introduces a *where-clause*. |

| Keyword | Value | Natural Data Format/Length | Explanation |
|---|---|---|---|
| FMDATE<br>or<br>FMDD | *date-from* | A10 | Only applies to programming objects.<br><br>The start date of a time period: All objects which were saved or cataloged on or after the specified date are processed.<br><br>If no end date is specified with TODATE , all objects from the specified date are selected for processing.<br><br>A date must be specified according to the setting of the DTFORM profile parameter (see *DTFORM - Date Format* in the *Parameter Reference* documentation) as indicated in the upper right corner of a SYSMAIN menu screen. The default setting is the international format *YYYY-MM-DD* (*YYYY* = year, *MM* = month, *DD* = day), for example, 2005-08-20. |
| FMTIME<br>or<br>FMTM<br>or<br>FMTT | *time-from* | A5 | Only applies to programming objects and if FMDATE is specified.<br><br>Specifies a start time: All objects which were saved or cataloged at or after the specified time (and date) are processed.<br><br>A time must be specified in the format *HH*:*II* (*HH* = hours, *II* = minutes), for example, 11:33. |
| FNR | *fnr* | N5 | The file number (FNR) of a source or a target system file (FNAT, FDIC or FUSER).<br><br>The source system file contains the object to be processed. The target system file is the system file to which the object is to be copied or moved, or where the object is renamed (or in the case of an error message, renumbered) if relevant.<br><br>Valid FNRs are 1 to 65535.<br><br>If no database ID (DBID) or FNR is specified and SYSMAIN is called with the system command SYSMAIN or the subprogram MAINUSER (see also ***Invoking SYSMAIN with Appl. Programming Interface***), the following applies: The DBID and FNR of the system file from which SYSMAIN was called are always used. For example: if you enter SYSMAIN from a library contained in the FUSER system file, the DBID and FNR of this file are used.<br><br>For rules and DDMs: The corresponding DIC specification can be used instead of FNR. If *fnr* is specified twice, the one specified last will be used. |

| Keyword | Value | Natural Data Format/Length | Explanation |
|---|---|---|---|
| HELP<br>or<br>? | - | - | Activates online **selective processing**.<br><br>You can either include the keyword HELP in the *with-clause* or enter a question mark (?) as the final character of an object name. |
| LANGUAGE | *language* | A9 | Only applies to error messages.<br><br>The code of the language of the error message to be processed.<br><br>The languages can be specified using any combination of language codes. For information on which language code is assigned to which language, see *Language Code Assignments* in **LANGUAGE* in the *System Variables* documentation. See also ***Specifying Languages***.<br><br>You can use an asterisk (*) to select all existing languages of the error messages to be processed. |
| LIBRARY<br>or<br>APPLIC<br>or<br>APL | *lib-name* | A8 | Only applies to programming objects, Predict sets, debug environments and error messages.<br><br>An optional keyword that indicates the name of a source or a target library. If you omit the keyword and respective value, the library where you logged on before you invoked SYSMAIN is used for processing.<br><br>The source library contains the object to be processed. The target library is the library to which the object is to be copied or moved, or where the object is renamed (or in the case of an error message, renumbered).<br><br>For system error messages, specify NATURAL-SYSTEM or NATURAL-SYS as *lib-name*.<br><br>*lib-name* must be specified immediately after the FROM and TO keywords. If the optional keyword LIBRARY is used, it must be entered between FROM or TO and *lib-name*. |
| MON<br>or<br>NOMON<br>or<br>MONOFF | - | - | Not applicable in batch mode.<br><br>Activates (MON) or deactivates (NOMON or MONOFF) tracing of the current activity in SYSMAIN. During processing, you are informed as to which object is being read, deleted, updated, added, and whether an error occurs. With programming objects, you are also informed about the action taken with the XRef data. This function is effective only with TP environments which can run in non-conversational mode. |

| Keyword | Value | Natural Data Format/Length | Explanation |
|---|---|---|---|
| NAME | *vsam-name* | A8 | The DDNAME/FCT entry for the source or target file number. |
| PROMPT or NOPROMPT | - | - | Not applicable in batch mode.<br><br>Enables (PROMPT) or disables (NOPROMPT) the SYSMAIN prompts. With NOPROMPT, no confirmation screen is displayed. For example, before any deletion, SYSMAIN prompts you for confirmation. |
| PASSWORD or PSW | *password* | A8 | The Adabas password of a source file and/or target system file which is used in a *where-clause*.<br><br>For rules and DDMs: The corresponding DIC specification can be used instead of PASSWORD. If *password* is specified twice, the one specified last will be used. |
| PROFILE | *name* | A8 | Only applies to profiles.<br><br>The name of the profile to be processed or a range of names. See also ***Specifying a Range of Names***. |
| RCOP | - | - | Specifies that a copy of the object being renamed is to be made. |
| REPLACE | - | - | Activates the replace option used in a *with-clause*.<br><br>An object with the same name in the target environment is replaced by the object to be processed.<br><br>**Note:** If a programming object is replaced it is also deleted from the Natural buffer pool; any existing cross-reference records are also deleted if Predict is installed. |
| RULE | *name* | A32 | Only applies to rules.<br><br>The name of the rule to be processed or a range of names. See also ***Specifying a Range of Names***. |
| SAVED | *name* | A9 | Only applies to programming objects.<br><br>The name of the saved (source) object to be processed or a range of names. See also ***Specifying a Range of Names***. |
| SEC | *dbid*<br>*fnr*<br>*password*<br>*cipher* | A80 | Not applicable to profiles and DL/I subfiles.<br><br>Specifies the environment of the FSEC source and/or target system file: database ID (*dbid*), file number (*fnr*), Adabas password (*password*) and Adabas cipher code (*cipher*). |
| SET | *Set-number* | N2 | Only applies to Predict sets.<br><br>The number of the Predict set or the start number of a range of Predict sets which are to be processed. |

| Keyword | Value | Natural Data Format/Length | Explanation |
|---|---|---|---|
| SETNO | *set-number* | N2 | Only applies to programming objects.<br><br>The number of the retained Predict set created with the Predict XRef save set option of the LIST XREF command. You can apply all SYSMAIN processing functions to the objects included in this set.<br><br>If any valid number is specified, SYSMAIN assumes a Predict set. If no number is specified, normal object processing is assumed. |
| SETLIBRARY | *set-library* | A8 | Only applies to programming objects.<br><br>Activates the option to overwrite the library specification for a Predict set as a part of the security for Predict files.<br><br>SETLIBRARY is only evaluated if a valid number has been specified for SETNO. |
| SETUSER | *set-user* | A8 | For programming objects:<br><br>Activates the option to overwrite the user ID specification for a Predict set as a part of the security for Predict files.<br><br>SETUSER is only evaluated if a valid number has been specified for SETNO.<br><br>For Predict sets:<br><br>The user ID specification of the Predict sets to be processed. When applied to the source specification, it is used as selection criterion. When applied to the target specification, it is used as new value.<br><br>It is not possible to specify ranges. |
| STOWED<br>or<br>BOTH | *name* | A9 | Only applies to programming objects.<br><br>The name of an object (or a range of names) for which the saved (source) *and* the cataloged object are to be processed (see also ***Specifying a Range of Names***). Only an object that exists as both a saved object *and* a cataloged object is processed.<br><br>The exceptions to this are copycode, text and recording, neither of which can be cataloged. However, they are included in processing when this option is specified. |
| THRU | *number*<br>or<br>*new-number* | N4 | Only applies to error messages.<br><br>The end number of a range of error message numbers to be processed if a start number is specified with AS . |

| Keyword | Value | Natural Data Format/Length | Explanation |
|---|---|---|---|
| TID | *terminal-ID* | A8 | Only applies to programming object.<br><br>A terminal ID: All objects that were saved or cataloged on the specified terminal are processed. |
| TO | *lib-name* | A8 | For programming objects, Predict sets, debug environments and error messages:<br>Specifies a target library.<br><br>For profiles, rules, DDMs and DL/I subfiles: Introduces a *where-clause*. |
| TODATE<br>or<br>TODD | *date-to* | A10 | Only applies to programming objects.<br><br>The end date of a time period: All objects which were saved or cataloged on or before the specified date are processed. A start date can be specified with FMDATE .<br><br>A date must be specified according to the setting of the DTFORM profile parameter (see *DTFORM - Date Format* in the *Parameter Reference* documentation) as indicated in the upper right corner of a SYSMAIN menu screen. The default setting is the international format *YYYY-MM-DD* (*YYYY* = year, *MM* = month, *DD* = day), for example, 2005-08-20. |
| TOSET | *Set-number* | N2 | Only applies to Predict sets.<br><br>The last number of a range of Predict sets which are to be processed. |
| TOTIME<br>or<br>TOTT<br>or<br>TOTM | *time-to* | A5 | Only applies to programming objects and if TODATE is specified.<br><br>The end time of a time period: All objects which were cataloged or saved at or before the specified time (and date) are processed.<br><br>The time must be specified in the format *HH*:*II* (*HH* = hours, *II* = minutes), for example, 11:33. |
| TYPE | *type* | - | The type of **programming object**, **error message**, **profile**, **rule** or **DL/I subfile** to be processed as listed in ***TYPE Specification*** below. |
| USER<br>or<br>USR | *user-id* | A8 | Only applies to programming objects.<br><br>A user ID: All objects that were saved or cataloged by the specified user are processed. |

| Keyword | Value | Natural Data Format/Length | Explanation |
|---|---|---|---|
| WHERE | *where-clause* | - | An optional keyword that indicates the start of a *where-clause*.<br><br>The *where-clause* must always follow the FROM or TO keyword and *lib-name* (if relevant); the sequence of the keywords and values within the clause can be specified in any order.<br><br>For details, see the direct command syntax in the object-type specific sections of the *SYSMAIN Utility* documentation. |
| WITH | *with-clause* | - | An optional keyword that indicates the start of a *with-clause*.<br><br>The keywords and values of the *with-clause* can be specified in any order, and the *with-clause* can be placed in any location within the direct command string, except in the first three positions.<br><br>For details, see the direct command syntax in the object-type specific sections of the *SYSMAIN Utility* documentation. |
| XREF | F<br>or<br>N<br>or<br>S<br>or<br>Y<br>or<br>D | A1 | Only applies to programming objects.<br><br>Indicates whether cross-reference (XRef) data stored on Predict system files is to be processed.<br><br>You can specify one of the following values:<br><br>F: All XRef data is processed and the object must be documented in Predict.<br><br>N: XRef data is not processed, except when using the DELETE command. If a cataloged object is deleted or replaced, SYSMAIN always deletes any existing XRef data for this object.<br><br>S: A specified object is processed regardless of whether it has XRef data or not.<br><br>Y: All XRef data is processed.<br><br>D: The object must be documented in Predict. Any existing XRef data is processed. |

| Keyword | Value | Natural Data Format/Length | Explanation |
|---|---|---|---|
| | | | For further details, see *XRef Considerations*. |

## TYPE Specification - Programming Objects

### Natural Data Format/Length: A20

The following table lists all valid object-type codes for programming objects:

| Code | Object Type |
|---|---|
| P | Program |
| N | Subprogram |
| S | Subroutine |
| M | Map |
| H | Helproutine |
| 0 | ISPF macro |
| 3 | Dialog |
| 5 | Processor |
| A | Parameter data area |
| G | Global data area |
| L | Local data area |
| C | Copycode |
| T | Text |
| R | Report |
| Z | Recording |
| 4 | Class |
| 7 | Function |
| 8 | Adapter |
| 9 | Resource |
| * | All programming object types |

## TYPE Specification - Error Messages

### Natural Data Format/Length: A1

The following table lists all valid type codes for error messages:

| Code | Type |
|---|---|
| S | Short error message |
| E | Extended (long) error message |
| A | All error message types: short and/or extended messages |

## TYPE Specification - Profiles

### Natural Data Format/Length: A3

The following table lists all valid type codes for profiles:

| Code | Type |
|---|---|
| E | Editor profile |
| D | Device profile |
| M | Map profile |
| P | Parameter profile |
| * | All profile types. |

## TYPE Specification - Rules

### Natural Data Format/Length: A2

The following table lists all valid type codes for rules:

| Code | Type |
|---|---|
| A | Automatic rule |
| F | Free rule |
| AF | All rule types: automatic and/or free rules. |

### TYPE Specification - DL/I Subfiles

**Natural Data Format/Length: A1**

The following table lists the valid type codes for DL/I subfiles:

| Code | Type |
|---|---|
| D | NDBs and UDFs |
| P | NSBs |

## Specifying a Range of Names

All SYSMAIN functions provide the option to specify either a name or a range of names for the objects to be processed. In addition, in menu mode, on a **Find Selection** or **List Selection** screen, you can specify a name or a range of names to limit the number of objects displayed. See also ***To shorten a selection list*** in *Using a Selection List*.

When using the find or the list function with programming objects, you can also specify a range of library names. The same applies when using the list function with debug environments or the find function with error messages. However, specifying library ranges may have a negative effect on the response time depending on how often the selection criteria occur.

The valid notations for name ranges are listed below where *value* denotes any combination of one or more characters:

| Input | Items Selected |
|---|---|
| * | All items. |
| *value** | All items with names that start with *value*.<br><br>Example: AB*<br>Selected: AB, AB1, ABC, ABEZ<br>Not selected: AA1, ACB |
| *value*> | All items with names greater than or equal to *value*.<br><br>Example: AB><br>Selected: AB, AB1, BBB, ZZZZZZZ<br>Not selected: AA1, AAB |
| *value*< | All items with names less than or equal to *value*.<br><br>Example: AX<<br>Selected: AB, AWW, AX<br>Not selected: AXA, AY |

# 95 Special Commands Issued to SYSMAIN

There are commands that can be issued to the SYSMAIN utility to perform functions related to the operation of the utility itself or to define the security for Natural system files.

| Command | Function |
|---|---|
| ADAON<br>or<br>NOADA<br>or<br>ADAOFF | Activate (ADAON) or deactivate (NOADA or ADAOFF) error trapping.<br><br>Trap abnormal database errors (only applicable online with programming objects) for debugging purposes. |
| BATCH<br>or<br>NOBATCH | Switch the SYSMAIN utility into batch mode (BATCH), whereby all processing is done as if SYSMAIN was running in batch. If online automated processing is used, a batch report is then displayed: see also ***Online Report Mode***.<br><br>NOBATCH switches the SYSMAIN utility back to online mode. |
| CLEAR | Clear the current work area. This function can be useful if the source code of a large object is contained in the work area and the SYSMAIN utility therefore requires a larger ESIZE. |
| DISPLAY | Display the extended (long) message text for the error which has occurred. |
| FINDFIRST | Stop the FIND command for programming objects when the first library is found that contains the specified object. |
| FINDALL | Cause the FIND command for programming objects to search in all libraries. |
| MON<br>or<br>NOMON<br>or<br>MONOFF | Activate (MON) or deactivate (NOMON or MONOFF) tracing of the current activity in SYSMAIN. See also MON in *Keywords and Variables in Direct Commands*. |
| PROMPT<br>or<br>NOPROMPT | Enable (PROMPT) or disable (NOPROMPT) the SYSMAIN prompts. See also PROMPT in *Keywords and Variables in Direct Commands*. |
| SET | Invoke the **Command Help** window where all special SYSMAIN commands are explained. |

| Command | Function |
|---|---|
| SET FDIC | Invoke a window to specify Adabas security information for the Predict system file. This refers to the profile parameter FDIC (see *FDIC - Predict System File* in the *Parameter Reference* documentation). In batch mode, you can specify security information by using the keyword DIC in a *where-clause* as indicated in the direct command syntax in the object-type specific sections of the *SYSMAIN Utility* documentation.<br><br>See also ***Special Considerations for Administrators***. |
| SET FNAT | Invoke a window to specify Adabas security information for the SYSMAIN source and target system files. In batch mode, you can specify security information by using the keyword SEC in a *where-clause* as indicated in the direct command syntax in the object-type specific sections of the *SYSMAIN Utility* documentation.<br><br>See also ***Special Considerations for Administrators***. |
| SET FSEC | Invoke a window to specify Adabas security information of the FSEC system file if Natural Security is installed. This refers to the profile parameter FSEC (see *FSEC - Natural Security System File* in the *Parameter Reference* documentation).<br><br>See also ***Special Considerations for Administrators***. |
| SET PC | Only applies if Natural Connection and Entire Connection are installed.<br><br>Activate the PC connection. This setting can be intermittently changed with the %+ and %- terminal commands (see also *Enable/Disable Use of Natural Connection* in the *Terminal Commands* documentation). SET PC then results in SYSMAIN re-verifying the status of the PC connection. |
| STATUS | Display the current values of SYSMAIN variables that are important for Software AG technical support. |
| TOTAL | Invoke the **Results of Function** window which verifies the processing of the last SYSMAIN function executed. The following information is displayed for saved (source) and cataloged objects:<br><br>**Read** — Total number of objects which were actually read, based on the object name specification.<br><br>**Rejected** — Total number of objects read which were then rejected, based on the selection criteria specified. See also ***Object Rejection and Reasons***.<br><br>**Processed** — Total number of objects which were processed.<br><br>**Added** — Total number of new objects added to the target environment.<br><br>**Updated** — Total number of existing objects updated. (Where possible, SYSMAIN attempts to update existing objects instead of deleting and adding new ones.)<br><br>**Deleted** — Total number of objects deleted from either the source or target environment, depending on the function and the setting of the replace option.<br><br>**Replaced** — Total number of objects which were replaced in the target environment. |

| Command | Function | |
|---|---|---|
| | **Not Repl.** | Total number of objects which were *not* replaced in the target environment. |
| | **Recs.Read:** | Total number of records which were read. |
| .<br><br>or<br>`END`<br><br>or<br>`QUIT` | Terminate SYSMAIN. | |

# 96 Processing Status and Error Notification


- Object Rejection and Reasons ..... 660
- Status Messages ..... 661
- SYSMAIN Error Notification ..... 665

This section describes possible reasons for object rejection during function processing, the status messages displayed after processing, and error notification during processing.

## Object Rejection and Reasons

If, during the execution of a SYSMAIN function, one or more objects were found to satisfy the specified selection criteria, but some or all of these objects were then rejected for further processing, any of the following Natural system errors occurs:

```
4867:Nothing found for this request.
4810:All data rejected by these selection criteria.
4893:Normal completion, but some data were rejected.
```

You can use the SYSMAIN command `TOTAL` (see *Special Commands issued to SYSMAIN*) to review the specific status of a request.

Possible reasons for object rejection are described in the following section:

- Invalid Object Type, Date/Time, User or Terminal
- Identical Target Name
- XREF and User Exits
- No Short or Extended Error Message
- Library Restrictions

### Invalid Object Type, Date/Time, User or Terminal

- An object was selected and then rejected because the object type was not valid for the type of processing specified. For example, when processing programming objects, all maps are rejected if processing has been restricted to objects of the type program or subroutine.
- An object was selected and then rejected because the date or time on which the specified object was saved or cataloged did not fall within the date or time range specified.
- An object was selected and then rejected because the user ID or terminal ID by which the specified object was saved or cataloged did not match with the user ID or terminal ID specified.
- A cataloged object was selected and then rejected because the profile parameter `RECAT` was set to `ON` and there was no saved (source) object corresponding to the cataloged object. See also: ***Using Profile Parameter RECAT***.
- A saved (only) object was selected and then rejected because the profile parameter `RECAT` was set to `ON` and the target environment already contained a cataloged object with the same name. See also: ***Using Profile Parameter RECAT***.

### Identical Target Name

- An object was selected and then rejected because the target environment already contained an object identified by the same name, and the replace option was not activated (**Replace** field set to `N` or keyword `REPLACE` not specified).
- A cataloged programming object of type `S` (subroutine) was selected and then rejected because the name of the external subroutine was identical to the name of another subroutine in the target library.
- A cataloged programming object of type `4` (class) was selected and then rejected because the name of the external class was identical to the name of another class in the target library.

### XREF and User Exits

- The XREF option was activated (**XREF** field *not* set to `N` or keyword `XREF` specified) and there were no XRef data for the programming object specified.
- A user exit routine was active and a non-zero return code was returned during processing of the object.

### No Short or Extended Error Message

- An extended (long) error message was selected and then rejected because there was no corresponding short error message in the source library.
- An extended (long) error message was selected but could not be processed because there was no corresponding short error message in the target environment.
- A short error message was selected to be moved, deleted or renumbered, but could not be processed because the corresponding extended error text was not included in the selection criteria. An extended error message must always have a corresponding short error message.

### Library Restrictions

- A library was specified which is controlled by Predict Application Control/Predict Application Audit, and the object cannot be handled by SYSMAIN.

## Status Messages

The table below describes the status messages that can be displayed in the **Message** column of a **Selection** screen or in a batch report (see also ***Online Report Mode***) after a SYSMAIN function was performed on an object. For the line commands mentioned, refer to *Selection Lists* in the object-type specific sections of the *SYSMAIN* documentation.

| Message | Explanation | Function completed? |
|---|---|---|
| `Class Exists` | Only applies to programming objects.<br><br>The cataloged object of a class had an external class name already used by another cataloged object of a class in the target environment. | No |
| `Copied` | One of the following line commands was executed from a **Copy/Rename Selection** screen:<br>`A`, or<br>`A`, `C` or `S` for programming objects, or<br>`A`, `E` or `S` for error messages, or<br>`A`, `D`, `E`, `M` or `P` for profiles. | Yes |
| `DB Error: nnn` | A database error was returned for the object during processing. | No |
| `Deleted` | One of the following line commands was executed from a **Delete Selection** screen:<br>`A`, or<br>`A`, `C` or `S` for programming objects, or<br>`A`, `E` or `S` for error messages, or<br>`A`, `D`, `E`, `M` or `P` for profiles. | Yes |
| `Dev exists` | Only applies to profiles.<br><br>The replace option was set to `N` and line command `A` or `D` was executed from a **Copy/Move/Rename Selection** screen. The device profile with the same name in the target environment was *not* replaced. | No |
| `Directory` | Line command `I` (display directory) was executed from a **Selection** screen. | Yes |
| `Edt exists` | Only applies to profiles.<br><br>The replace option was set to `N` and line command `A` or `E` was executed from a **Copy/Move/Rename Selection** screen. The editor profile with the same name in the target environment was *not* replaced. | No |
| `Err: NAT2999` | Only applies to programming objects.<br><br>A cataloged object was being processed with the XREF option set to `F`. No Predict entry exists on the FDIC system file specified for the object. | No |
| `Err: NAT4852` | A Natural Security violation occurred. | No |
| `Exit: nnn` | A user exit routine was active and a non-zero return code was returned by the exit (`nnn` = the return code). See also ***User Exit Routines***. | No |
| `Exported` | Only applies to programming objects and DDMs.<br><br>Line command `X` (PC download) was executed on a source object from a **Selection** screen. | Yes |
| `Ext Exists` | Only applies to error messages.<br><br>An attempt has been made to delete only the short error message of an error message for which an extended (long) error exists. This would have resulted | No |

| Message | Explanation | Function completed? |
|---|---|---|
| | in an extended (long) error message with no corresponding short message. This is not permitted in Natural. | |
| `File Listed` | Only applies to rules.<br><br>Line command `F` (display DDMs/fields) was executed on an automatic rule from a **Selection** screen. | Yes |
| `In Use` | An Adabas response code 145 was returned during Natural `UPDATE/READ` processing of an object. | No |
| `Invalid` | Only applies to debug environments.<br><br>An invalid line command was entered for one of the debug environments listed on a **Selection** screen. | No |
| `Listed` | Not applicable to DDMs and debug environments.<br><br>Line command `L` (display source code) was executed from a **Selection** screen. | Yes |
| `Map exists` | Only applies to profiles.<br><br>The replace option was set to `N` and line command `A` or `M` was executed from a **Copy/Move/Rename Selection** screen. The map profile with the same name in the target environment was *not* replaced. | No |
| `Moved` | One of the following line commands was executed from a **Move Selection** screen:<br>`A`, or<br>`A`, `C` or `S` for programming objects, or<br>`A`, `E` or `S` for error messages, or<br>`A`, `D`, `E`, `M` or `P` for profiles. | Yes |
| `Must be Auto` | Only applies to rules.<br><br>Line command `F` (display DDMs/fields) was executed on a free rule from a **Selection** screen. Line command `F` can only be executed on automatic rules. | No |
| `Name Erro` | Only applies to debug environments.<br><br>The rename function was used, but the new name specified was found to be invalid. Either no new name was specified for the selection, or the specified name either contained invalid special characters or did not start with an alphabetic character. | No |
| `NBP Deleted`<br><br>or<br><br>`Ignored` | Only applies to programming objects.<br><br>Line command `B` (delete object from buffer pool) was executed from a **Selection** screen. The message returned upon completion of processing is either `NBP Deleted` or `Ignored`, depending on whether deletion from the buffer pool for the specified object(s) has been confirmed or not. | Yes/No |

| Message | Explanation | Function completed? |
|---|---|---|
| No Lang 1 | Only applies to error messages.<br><br>Only language code 1 (English) is available for Natural system extended (long) error messages. An attempt has been made to copy a Natural system extended error message, and language code 1 has not been specified as a language. | No |
| no Short Err | Only applies to error messages.<br><br>An extended (long) error message was selected for further processing, but the target error message number has no corresponding short error message. | No |
| No Xref | Only applies to programming objects.<br><br>A cataloged object was being processed and the XREF option was set to Y or F. No XRef data exists on the FDIC system file specified for the object. | No |
| Not Found | Only applies to error messages and profiles.<br><br>An error in the update logic occurred during processing and the requested error message or profile could not be found. This implies that the specified error message or profile was deleted in the time between selection and update. | No |
| Not Replaced<br><br>or<br><br>Not Repld | The replace option was set to N and one of the following line commands was executed from a **Copy/Move/Rename Selection** screen:<br><br>A, or<br>A, C or S for programming objects, or<br>A, E or S for error messages, or<br>A, D, E, M or P for profiles.<br><br>The object with the same name in the target environment was *not* replaced. | No |
| Parm exists | Only applies to profiles.<br><br>The replace option was set to N and line command A or P was executed from a **Copy/Move/Rename Selection** screen. The parameter profile with the same name in the target environment was *not* replaced. | No |
| Printed | Only applies to programming objects and rules.<br><br>Line command H (produce hardcopy) was executed from a **Selection** screen. | Yes |
| Renamed<br><br>or<br><br>Copied | Only applies to debug environments and error messages.<br><br>One of the following line commands was executed from a **Rename Selection** screen: A for debug environments, or A, E or S for error messages.<br><br>The message returned upon completion of processing is either Renamed or Copied depending on whether the option to retain the original object was specified. | Yes |

| Message | Explanation | Function completed? |
|---|---|---|
| `Renamed as`<br><br>or<br><br>`Copied as` | Only applies to programming objects.<br><br>Line command `A`, `C` or `S` was executed from a **Rename Selection** screen. The message returned upon completion of processing is either `Renamed as` or `Copied as`, depending on whether the option to retain the original object was specified. | Yes |
| `Replaced` | The replace option was set to `Y` and one of the following line commands was executed from a **Copy/Move/Rename Selection** screen:<br><br>`A`, or<br>`A`, `C` or `S` for programming objects, or<br>`A`, `E` or `S` for error messages, or<br>`A`, `D`, `E`, `M` or `P` for profiles.<br><br>The object with the same name in the target environment was replaced. | Yes |
| `Sized` | Only applies to programming objects.<br><br>Line command `Z` (calculate size) was executed from a **Selection** screen. | Yes |
| `Subrtn Exists` | Only applies to programming objects.<br><br>The cataloged object of a subroutine had an external subroutine name already used by another cataloged subroutine in the target environment. | No |
| `Updated` | Only applies to error messages.<br><br>The text in the specified language did not previously exist for the error message selected, and SYSMAIN has updated the error message with the new language text. | Yes |
| `Src locked` | Only applies to programming objects.<br><br>Line command `A` or `S` was executed from a **Copy/Move/Rename/Delete Selection** screen.<br><br>The object was *not* processed because the source of the respective object was being locked. | No |

## SYSMAIN Error Notification

SYSMAIN always attempts to recover in the event of a runtime error during processing. This feature is automatically activated and uses the system variable `*ERROR-TA` (see also **ERROR-TA* in the *System Variables* documentation). This feature is deactivated when SYSMAIN is terminated normally.

If a terminal command (see also *%% and %. - Interrupt Current Operation* in the *Terminal Commands* documentation) is used to terminate SYSMAIN, this is considered an abnormal termination, and

the `*ERROR-TA` system variable is not reset. It can be reset by re-invoking SYSMAIN and terminating it normally. In the event that you have set the `*ERROR-TA` system variable, SYSMAIN resets it to its previously assigned value upon termination.

If invalid data has been specified with respect to the selection criteria, an error message is displayed in the message line. If you are uncertain as to the meaning of the short error message, the special command `DISPLAY` (see also *Special Commands Issued to SYSMAIN*) can be entered to activate a display of the corresponding extended (long) error message text.

This section covers the following topics:

- Data Entry Errors
- Processing Errors

### Data Entry Errors

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

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

### Processing Errors

If you make a request which causes a processing error, SYSMAIN issues an error report as shown in the example below:

```
16:51:08          *** SYSMAIN Error Report ***         2005-08-10

The following internal error occurred while processing the
SYSMAIN function xxxxxx (cc):

   Error in field specification for IF SELECTION statement.


Error Number .. eeee
Program ....... pppppppp
Status Code ... s          Status ...... tttttttt
Line .......... llll       Level ....... vv
Device ........ dddddddd
User ID ....... uuuuuuuu   User Name ... nn...nn
```

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

The values in the window above have the following meanings:

| Field | Value | Explanation |
|---|---|---|
| **SYSMAIN function** | *xxxxxx* | The SYSMAIN function that is being performed. |
| | *cc* | An internal status code useful for Software AG technical support. The following codes can be displayed:<br><br>A Automatic processing.<br>D XRef data is being deleted.<br>E Error in processing (flag for SYSMAIN).<br>F Status setting when XRef data is being processed.<br>G Status setting when XRef data is being processed.<br>H Selection list processing.<br>I Option is being processed.<br>S Single object processing.<br>T SYSMAIN termination by command processor.<br>V Status setting when XRef data is being processed.<br>X SYSMAIN termination by command processor.<br>Y Validation error has occurred, redisplay should follow.<br>Z Validation error has occurred, redisplay should follow. |
| **Error Number** | *eeee* | Corresponds to the system variable `*ERROR-NR` (see the *System Variables* documentation). |
| **Program** | *pppppppp* | Corresponds to the system variable `*PROGRAM` (see the *System Variables* documentation). |
| **Status Code** | *s* | The kind of error. Possible codes are:<br><br>C Command processing error.<br>L Logon error.<br>O Object time error.<br>S Non-correctable syntax error. |
| **Line** | *llll* | Corresponds to the system variable `*ERROR-LINE` (see the *System Variables* documentation). |
| **Device** | *dddddddd* | Corresponds to the system variable `*DEVICE` (see the *System Variables* documentation). |
| **User ID** | *uuuuuuuu* | Corresponds to the system variable `*USER` (see the *System Variables* documentation). |
| **Status** | *tttttttt* | Corresponds to the system variable `*ERROR-TA` (see the *System Variables* documentation). |
| **Level** | *vv* | Corresponds to the system variable `*LEVEL` (see the *System Variables* documentation). |
| **User Name** | *nn...nn* | Corresponds to the system variable `*USER-NAME` (see the *System Variables* documentation). |

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

If the special command `DISPLAY` is entered in the window (see also *Special Commands Issued to SYSMAIN*), the extended (long) message text for the error incurred is displayed.

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

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

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

# 97 Special Considerations for Administrators


- File Security ..... 670
- Natural Security ..... 671
- User Exit Routines ..... 672

This section describes the security aspects of the SYSMAIN utility and the user exit routines supplied for SYSMAIN.

## File Security

The file security (that is, passwords and cipher codes) relates to the security that has been defined for a system file in an Adabas or a VSAM environment. If file security has been defined for a system file, you need to specify a password, cipher code and/or VSAM name for the source and/or target system file required before you perform a SYSMAIN function. Otherwise, Adabas or VSAM will issue an appropriate error message. You do not have to provide security information for the default system files assigned to the Natural session at the start of the SYSMAIN utility.

### To specify passwords and cipher codes

1 From any SYSMAIN utility menu, invoke a security window for the required system file by using either a PF key or a special command as indicated in the table below:

| System File | Command | PF Key | Objects/Data Affected |
|---|---|---|---|
| FUSER, FNAT | `SET FNAT` | PF12 | - programming objects<br>- debug environments<br>- error messages<br>- profiles |
| FDIC | `SET FDIC` | PF11 | - rules<br>- DL/I subfiles<br>- DDMs<br>- XRef information |
| FSEC | `SET FSEC` | PF10 | - Natural Security profile |

The security window that appears for the specified system file looks similar to the example of the FUSER and FNAT system files below:

```
+-----------------------------------------------------+
!   --- Security for the Natural System Files ---    !
!                                                     !
!  Specify the password(s), cipher(s) and VSAM FCT   !
!  name(s) for the source/target file(s) below:      !
!                                                     !
!      - Source -                - Target -           !
! Library .... OLDLIB       Library .... NEWLIB      !
! Database ... 10           Database ... 10          !
! File ....... 32           File ....... 32          !
!                                                     !
! Password ...              Password ...             !
! Cipher .....              Cipher .....             !
! VSAM Name .. ________     VSAM Name .. ________    !
+-----------------------------------------------------+
```

2 In the window, enter the appropriate password(s), cipher code(s) and/or VSAM name for the required source and/or target system file.

**Note:** The **Library** field is applicable only when processing programming objects, debug environments or error messages.

Once file security is defined, the SYSMAIN utility uses this security information for all subsequent processing. If you then require that the default security information (obtained at the initialization of the session) be used, you must re-invoke the corresponding security window and clear the password, cipher code and/or VSAM name fields. The passwords and cipher codes are non-display, so even though the fields *appear* to be empty, they should be cleared again.

## Natural Security

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

- Defining the Natural Security Environment

- Restricting Use of SYSMAIN under Natural Security

### Defining the Natural Security Environment

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

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

By default, the current FSEC settings assigned at the start of the Natural session are used. If you change these settings (in the window **Security for Natural Security (FSEC) Files**), they remain in effect until they are changed by the next `SET FSEC` process. In batch mode or direct command mode, the `SEC` keyword should be used to specify the file security and assignments of the request.

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

### Restricting Use of SYSMAIN under Natural Security

The use of the SYSMAIN utility itself can be restricted, or the use of the source and target libraries to be handled with the SYSMAIN utility can be restricted. The use of SYSMAIN functions when invoked via the application programming interface MAINUSER can be controlled separately. See *Protecting Utilities* in the *Natural Security* documentation for details.

## User Exit Routines

The user exit routines of the SYSMAIN utility are used to provide information on each object being processed or control function processing. A user exit routine is a Natural subprogram, which is invoked with a `CALLNAT` statement.

The source codes of the subprograms and the data areas they use are stored as source objects under the names SM-UX-*nn* (*nn* = 01 to 11) in the library SYSMAIN. To make a user exit routine available, you have to catalog the corresponding source object under the name MAINEX*nn*, either in the library SYSMAIN or in one of its steplibs.

**Note:** The names of source objects and cataloged objects of user exit routines are different to ensure that the overwriting of the source objects by an update installation does not affect the cataloged objects.

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

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

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

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

The individual user exit routines are described in the following section:

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

### MAINEX01 - First User Exit Routine for Object Interrogation

| | |
|---|---|
| Function | Interrogate the current value settings of the data elements associated with an object *before* the object is processed by SYSMAIN. |
| Remarks | Any object passed to MAINEX01 can be rejected by setting the `RESP-CODE` parameter to a non-zero value.<br><br>If any additional logic is to be performed, the transaction may *not* be at end-of-transaction status and so no `END TRANSACTION` or `BACKOUT TRANSACTION` statement should be issued.<br><br>Control must be returned to SYSMAIN. |
| Parameters | `PARM-AREA1 (A250)` SYSMAIN parameter area (fixed values). |

| | |
|---|---|
| | `PARM-AREA2 (A250)` SYSMAIN parameter area (variable values).<br>`RESP-CODE (B1)` Response code to be returned to SYSMAIN.<br>**Note:** Only the `RESP-CODE` parameter can be modified. |
| Local Data Area | SM-UX-L |

### MAINEX02 - Second User Exit Routine for Object Interrogation

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

### MAINEX03 - User Exit Routine for Request Interrogation

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

### MAINEX04 - User Exit Routine for Modification of File Assignments

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

### MAINEX05 - User Exit Routine for Verification of Direct Commands

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

### MAINEX06 - User Exit Routine for SYSMAIN Initialization

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

| | |
|---|---|
| Parameter Data Area | SM-UX-L6 |

### MAINEX07 - User Exit Routine for SYSMAIN Termination

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

### MAINEX08 - User Exit Routine for Nothing Found in Batch Mode

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

### MAINEX09 - User Exit Routine for Abnormal Termination in Batch Mode

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

### MAINEX10 - User Exit Routine for Command Errors in Batch Mode

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

### MAINEX11 - User Exit Routine for Setting Special Flags to SYSMAIN

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

# XVIII SYSNCP Utility

# 98 SYSNCP Utility

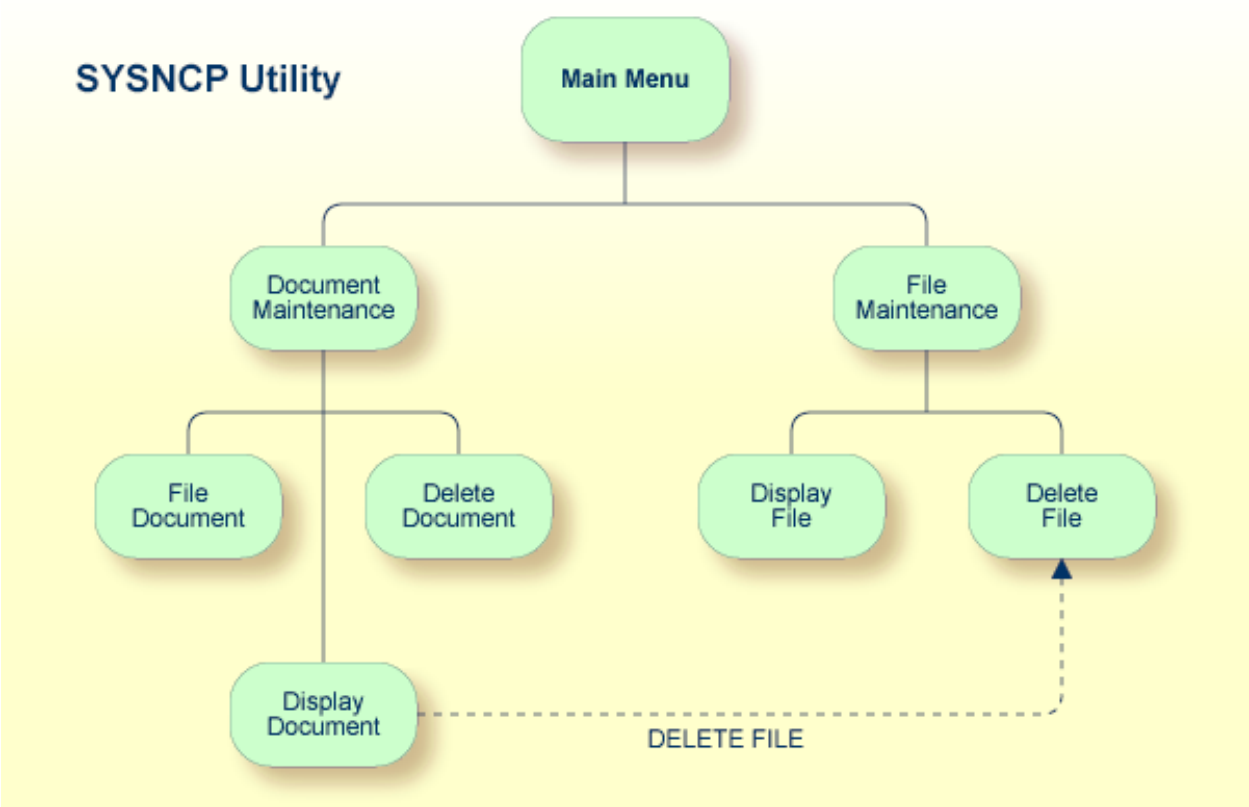

- Introducing the SYSNCP Utility ..... 682
- Invoking SYSNCP ..... 688
- Processor Selection ..... 689
- Header Records ..... 690
- Keyword Maintenance ..... 700
- Function Maintenance ..... 705
- Runtime Actions ..... 710
- Processor Cataloging ..... 715
- Administrator Services ..... 715
- Session Profile ..... 723

The utility SYSNCP is used to define command-driven navigation systems for Natural applications.

The Natural Command Processor (NCP) consists of two components: maintenance and runtime. The utility SYSNCP is the maintenance part which comprises all facilities used to define and control navigation within an application. The PROCESS COMMAND statement (see the *Statements* documentation) is the runtime part used to invoke Natural programs.

## Introducing the SYSNCP Utility

Applications which enable users to move from one activity to another activity by using direct commands far exceed in usability the ones which force the user to navigate through menu hierarchies to a desired activity.




The figure above illustrates the advantage of using direct commands. In an application in which menu hierarchies form the basis for navigation, a user wishing to advance from the Display Document facility to the Delete File facility would have to return to the Main Menu via the document branch and then enter the file branch. This is clearly less efficient than accessing the Delete File facility directly from the Display Document facility.

Below is information on:

- Object-Oriented Data Processing
- Features of the Command Processor
- Components of the Command Processor
- What is a Command?
- Creating a Command Processor


## Object-Oriented Data Processing

The Natural command processor is used to define and control navigation within an application. It could be used, for example, to define a command DISPLAY DOCUMENT to provide direct access to the Display Document facility. When a user enters this command string in the Command line of a screen (for which this command is allowed), the Natural command processor processes the input and executes the action(s) assigned to the command.

In contrast to menu-driven applications, the command-driven applications implemented with the Natural command processor take a major step toward object-oriented data processing. This approach has the following advantages:

- The design of an application need not depend on the way in which a certain result can be reached, but only on the desired result itself. Thus, the design of an application is no longer influenced by the process flow within its components.
- The processing units of an application become independent of one another, making application maintenance easier, faster and much more efficient.
- Applications can be easily expanded by adding independent processing units. The resulting applications are, therefore, not only easy to use from an end-user's view, but also easier to create from a programmer's view.

The Natural command processor has the following additional benefits:

- **Less Coding**
  Instead of having to repeatedly program lengthy and identically structured statement blocks to handle the processing of commands, you only have to specify a PROCESS COMMAND statement that invokes the command processor; the actual command handling need no longer be specified in the source code. This considerably reduces the amount of coding required.
- **More Efficient Command Handling**
  As the command handling is defined in a standardized way and in one central place, the work involved in creating and maintaining the command-processing part of an application can be done much faster and much more efficiently.
- **Improved Performance**
  The Natural command processor has been designed with particular regard to performance aspects: it enables Natural to process commands as fast as possible and thus contributes to improving the performance of your Natural applications.

### Features of the Command Processor

The Natural command processor provides numerous features for efficient and user-friendly command handling:

- **Flexible Handling of Commands**
  You can define aliases (that is, synonyms for keywords), and abbreviations for frequently used commands.
- **Automatic Check for Uniqueness of Abbreviated Keywords**
  The command processor automatically compares every keyword you specify in SYSNCP with all other keywords and determines the minimum number of characters in each keyword required to uniquely identify the keyword. This means that, when entering commands in an application, users can shorten each keyword to the minimum length required by the command processor to distinguish it from other keywords.
- **Local and Global Validity of Commands**
  You can specify in SYSNCP whether the action to be performed in response to a specific command is to be the same under all conditions or situation-dependent. For example, you can make the action dependent on which program was previously issued. In addition, you can define a command to be valid under one condition but invalid under another.
- **Error Handling for Invalid Commands**
  You can attach your own error-handling routines to commands or have error input handled by Natural.
- **Functional Security**
  With Natural Security, library-specific and user-specific conditions of use can be defined for the tables generated with SYSNCP. Thus, for your Natural applications you can allow or disallow specific functions or keywords for a specific user. This is known as functional security. See also the section *Functional Security* in the *Natural Security* documentation.
- **Help Text**
  In SYSNCP, you can attach help text to a keyword or a command. Then, by specifying a PROCESS COMMAND ACTION TEXT statement, you can return command-specific help text to the program.
- **Online Testing of Command Processing**
  If the execution of a command does not produce the intended result, you can find out why the command was not processed correctly by using the PROCESS COMMAND statement (see the *Statements* documentation) and the EXAM* sample test programs (source form) provided in the Library SYSNCP. The endings of the EXAM-* program names appear as abbreviations at the top border line of the relevant action windows (for example, EXAM-C appears as **C**).

#### To test a command processor at runtime

1. Enter the direct command EXAM to list all test programs. The **Demonstrate PROCESS COMMAND Statement** window is displayed.

2 Enter Function Code **O** to open a processor.

3 Enter the name of the processor.

4 Choose any of the Functions Codes listed (for example, C for CHECK) to apply command actions.

5 Enter Function Code **Q** to close the processor.

### Components of the Command Processor

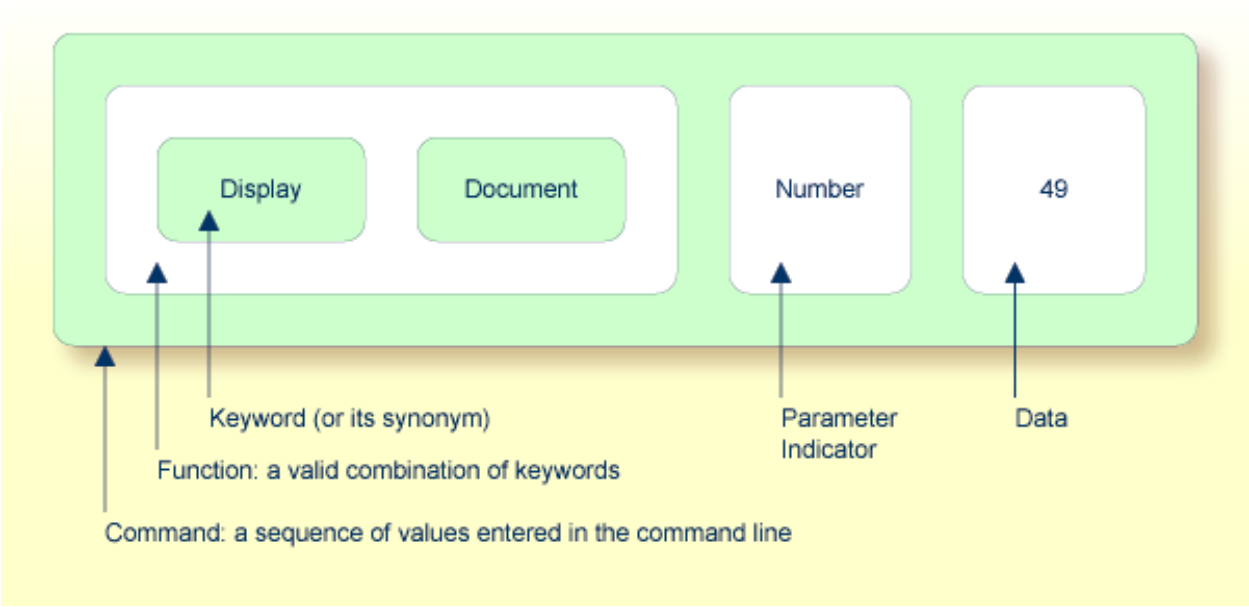
The Natural command processor consists of two parts: a development part and a runtime part:

- The development part is the utility SYSNCP, which is described in this section. With the utility SYSNCP you define commands (as described below) and the actions to be performed in response to the execution of these commands. From your definitions, SYSNCP generates decision tables which determine what happens when a user enters a command. These tables are contained in a Natural member of type Processor.
- The runtime part is the statement PROCESS COMMAND, which is described in the *Statements* documentation. This statement is used to invoke the command processor within a Natural program. In the statement, you specify the name of the processor to be used to handle the command input by a user at this point.

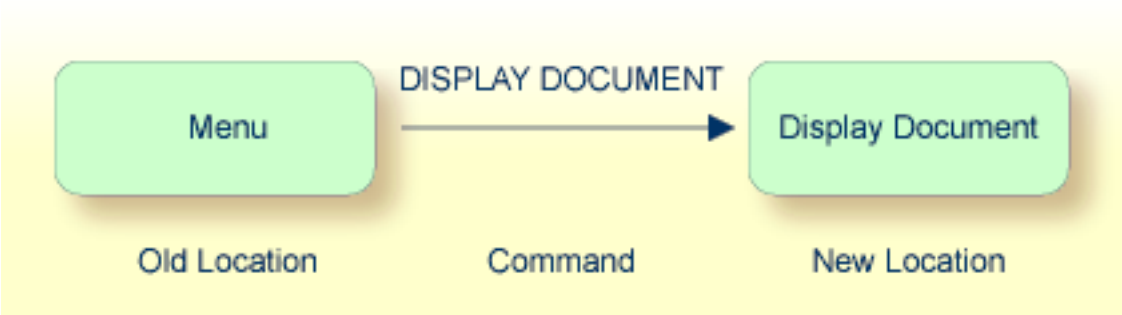
### What is a Command?

A command is any sequence of values entered in the Command line which is recognized and processed by an application. Commands can contain up to three elements:

- **Function:**
  One or more valid keywords. For example, MENU or DISPLAY DOCUMENT.
- **Parameter Indicator:**
  Optional. A keyword which introduces command data.
- **Command Data:**
  Information to be sent to a function. Command data can be alphanumeric or numeric, for example, the name or the number of the file to be displayed.




Commands are always executed from a situation within an application; the position where this situation is reached is referred to as a location. Commands take the user from one location to another location; thus, each command can be viewed as a vector:

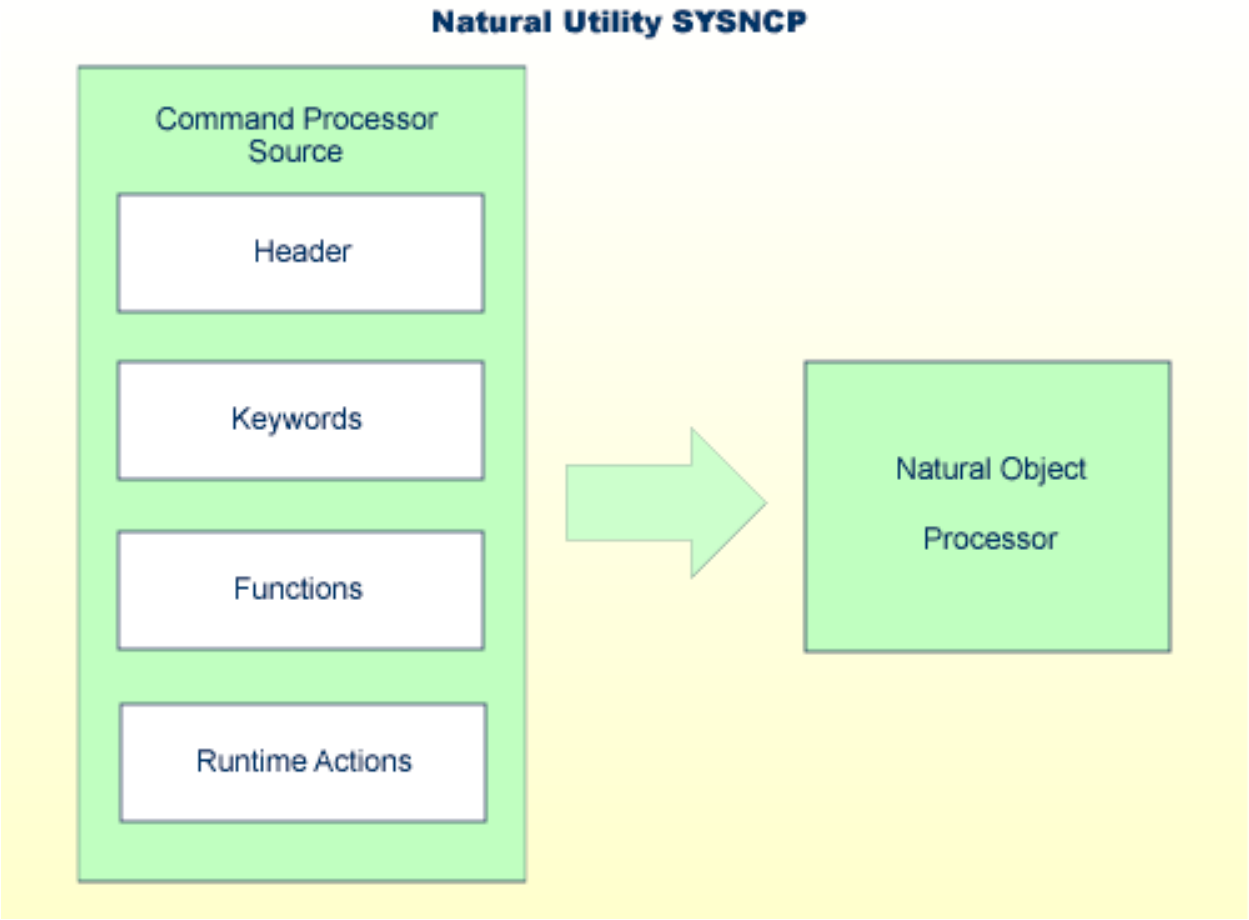



The location from which a certain command can be issued can be restricted on a system-wide or user-specific basis. On a system-wide basis, for example, the functions specified within commands can be local or global. A global function can be issued from *any* location while a local function can only be issued from specified locations. Restrictions can be placed on keywords and functions, however, if Natural Security is active in your environment.

### Creating a Command Processor

The utility SYSNCP is used to create and maintain command processors. A command processor contains decision tables which determine what happens when a user enters a valid command.

The creation of a command processor is a cumulative operation involving several steps, from header definition, which establishes general defaults for the processor, to keyword definition, function definition and the linking of actions to functions. Special editors are provided by SYSNCP for the purpose of specifying keywords, functions and actions.




The end product of command processor development is a complex command processor source, which, when cataloged, generates a Natural object of type Processor. Whenever this object is referenced by the Natural statement PROCESS COMMAND, the runtime system of the Natural command processor is triggered.

The following is a summary of the steps necessary to create a command processor.

**To create a command processor**

1 **Verify/Modify the Session Profile.**

SYSNCP itself uses a Session Profile which contains various parameters which control how SYSNCP is to perform certain actions and how information is to be displayed. Desired modifications can be made and the resulting profile can be saved with a given user ID. See the section ***Session Profile***.

2 **Initialize the Command Processor.**
The name of the command processor and the library into which it is to be stored are specified.

3 **Define Global Settings (Header).**
Various global settings for the command processor are defined. For example, descriptive text for keywords during editing, minimum and maximum length for keywords, in which sequence keywords are to be processed at runtime, runtime error-handling, and whether PF keys can be used at runtime to invoke functions. See the section ***Header Records***.

4 **Define Keywords.**
Each keyword which is to be processed by the command processor is defined together with an indication as to whether the keyword is to be entered as the first, second or third entry of a command. Keyword synonyms can also be defined as well as parameter indicators. User text can be defined for each keyword. This text can subsequently be read at runtime using the PROCESS COMMAND ACTION TEXT statement. See the section ***Keyword Maintenance***.

5 **Define Functions.**
Functions are defined by validating keyword combinations. A function can be defined as local (can only be invoked from a specific location within an application) and/or global (can be invoked from anywhere within an application). See the section ***Function Maintenance***.

6 **Define Runtime Actions.**
The actions to be taken by the command processor when a command is issued at runtime are specified. Example actions are: fetch a Natural program, place a command at the top of the Natural stack, place data at the top of the Natural stack, change contents of the Command line. See the section ***Runtime Actions***.

7 **Catalog Command Processor.**
The resulting source is cataloged as a Natural object (type Processor) in the designated Natural library. The command processor can now be invoked by a Natural program using the `PROCESS COMMAND` statement. See the section ***Processor Cataloging***.

## Invoking SYSNCP

### To invoke the SYSNCP utility

- Enter the system command SYSNCP.

  The Processor Source Maintenance menu is displayed:

```
18:22:53              ***** NATURAL SYSNCP UTILITY *****          2000-05-22
User SAG                 - Processor Source Maintenance -

                     Code  Function

                      S    Select Processor
                      N    Create New Processor
                      H    Modify Header
                      K    Define Keywords
                      F    Define Functions
                      R    Define Runtime Actions
                      C    Catalog Processor
                      A    Administrator Services
                      ?    Help
                      .    Exit

               Code .. _    Name .. SAGTEST_  Library .. SYSNCP__



Logon to SYSNCP accepted.
Command ===>
Enter-PF1---PF2---PF3---PF4---PF5---PF6---PF7---PF8---PF9---PF10--PF11--PF12---
      Help  Cmd   Exit  Last  List  Flip                                  Canc
```

From this menu, you can invoke all functions necessary to create and maintain a command processor. To invoke a function, enter the code letter in the Code field.

**Note:** When you invoke the SYSNCP utility or restart SYSNCP, the user exit NCP-USR1 is invoked for dynamic customization purposes: see the program NCP-USR1 delivered in the Natural system library SYSNCP.

### Help

For help on individual input fields (and also on some output fields) in SYSNCP, place the cursor on the field and press PF1.

## Processor Selection

The Select Processor function results in a list of all existing command processor sources with related information. If Natural Security is installed, only those sources are listed which can be cataloged to a library to which you are allowed to log on. These restrictions do not apply to those users who have administrator status.

### To invoke the Select Processor function

1 In the Processor Source Maintenance menu, enter Function Code **S**.

2 Press ENTER.

The following information is provided for each processor:

| | |
|---|---|
| Name | The name of the command processor. |
| Library | The name of the Natural library for which a processor is created. When the processor is cataloged, it is stored in this library. |
| User ID | The ID of the user who created the processor. |
| Date | The date the processor was created. |
| Status | The stage of development of the processor. For possible status values, see ***Current Status*** in the section *Header Records*. |
| Cat | Indicates if the processor has been cataloged. |

**Note:** With the user exit NCP-SELX (delivered in the Natural system library SYSNCP), you can limit the display to certain processors.

3 In the **Ac** field, enter any character to select a processor.

The Processor Source Maintenance menu is displayed, where the name of the selected processor is automatically placed in the Name field.

If you enter a question mark (?) in the Ac field, a window is be displayed, listing other possible options.

The name and library name of a command processor can be one to eight characters long. It can consist of upper-case alphabetical characters (A - Z), numeric characters (0 - 9) and the special characters: "-", "/ ", "$", "&", "#", "+" and "_".

## Header Records

The header maintenance facility defines various global settings for a command processor. These definitions are collectively referred to as a header. Seven header maintenance screens are provided for creating and modifying headers. Header settings for a command processor can be updated at any stage of development (see the following section). After the settings have been modified, the status of a command processor is always set to Header (see also ***Current Status***).

Below is information on:

- Create New Processor

- Modify Header - General Explanations
- Keyword Runtime Options - Header 1
- Keyword Editor Options - Header 2
- Miscellaneous Options - Header 3
- Command Data Handling - Header 4
- Runtime Error Handling - Header 5
- Statistics - Header 6
- Status - Header 7


## Create New Processor

### To create a new command processor

1. In the Processor Source Maintenance menu, enter Function Code **N** (Create New Processor), the name of the command processor to be created, and
   the name of the Natural library in which the command processor is to be later cataloged.
2. Press ENTER.

   The first header maintenance screen is displayed.

The first header maintenance screen and the following ones are filled with default values that can be edited.

## Modify Header - General Explanations

The Modify Header function is used to maintain an existing header; that is, to modify the various header settings for a given command processor.

### To modify an existing header

1. In the Processor Source Maintenance menu, enter Function Code **H** (Modify Header), the name of the corresponding command processor, and
   the name of the library into which this command processor has been cataloged.
2. Press ENTER.

   The first header maintenance screen is displayed.
3. Modify any input field in the header maintenance screens described below.
4. Press ENTER to confirm modifications.

Seven different screens are available for the definition and maintenance of a processor header (for the definition of a header, see the previous section).

### To navigate between the header maintenance screens

- Use PF8 (forward) or PF7 (backward).

Each of the screens contains the following information:

| | |
|---|---|
| Name | The name of the command processor. |
| Library | The name of the library into which the resulting command processor object is to be placed after being cataloged. |
| DBID, FNR | The database ID and file in which the specified library is located. |
| Created by | The user ID of the Natural user who initialized this command processor. |
| Date | The date the command processor was initially created. |
| Current Status | The command processor status:<br><br>Init — The command processor has been initialized.<br>Header — The header for the command processor has been created/modified.<br>Keysave — Keywords have been defined and saved.<br>Keystow — Keywords have been checked and stowed.<br>Function — Keyword combinations have been defined.<br>Action — Runtime actions have been defined.<br>Object — An object form of the command processor has been created.<br>Frozen — The command processor has been frozen.<br>Copied — The command processor has been copied.<br>Error — An error has been detected. |

## Keyword Runtime Options - Header 1

When you select the Modify Header function (as described above), the **Processor Header Maintenance 1** screen is displayed:

```
16:40:19                ***** NATURAL SYSNCP UTILITY *****          2000-05-04
User SAG                - Processor Header Maintenance 1 -

Modify Processor            Name SAGTEST  Library SYSNCP   DBID 10    FNR 32
Created by SAG      Date 2000-04-29                      Current Status Init

Keyword Runtime Options:
------------------------
   First  Entry used as ....... Action__________
   Second Entry used as ....... Object__________
   Third  Entry used as ....... Addition________

   Minimum Length ............. _1
   Maximum Length ............. 16
   Dynamic Length Adjustment .. -

   Keyword Sequence ........... 123_____
   Alternative Sequence ....... ________
   Local/Global Sequence ...... LG______

Processor Header with name SAGTEST for library SYSNCP has been added.
Command ===>
Enter-PF1---PF2---PF3---PF4---PF5---PF6---PF7---PF8---PF9---PF10--PF11--PF12---
      Help  Cmd   Exit  Last  List  Flip  -     +                       Canc
```

Various attributes which are to apply for the keywords defined for the command processor are entered on this screen.

| Field | Explanation |
|---|---|
| First Entry used as | A descriptive text which is to be associated with all keywords which are entered as the first entry (entry type 1) when defining a keyword sequence.<br><br>For example, if the first keyword of a keyword sequence is to represent the action to be performed (DISPLAY, DELETE, etc.), the descriptive text "Action" could be entered in this field.<br><br>The first four characters of the text entered in this field appear under the column heading **Use** in the Keyword Editor as described in the section *Keyword Maintenance*. |
| Second Entry used as | A descriptive text which is to be associated with all keywords which are entered as the second entry (entry type 2) when defining a keyword sequence.<br><br>If, for example, the second keyword of a keyword sequence is to represent the object to be used (DOCUMENT, FILE, etc.), the descriptive text "Object" could be entered in this field. |

| Field | Explanation |
|---|---|
| | The first four characters of the text entered in this field appear under the column heading **Use** in the Keyword Editor as described in the section *Keyword Maintenance*. |
| Third Entry used as | A descriptive text (TITLE, PARAGRAPH, etc.) which is to be associated with all keywords which are entered as the third entry (entry type 3) when defining a keyword sequence.<br><br>The first four characters of the text entered in this field appear under the column heading **Use** in the Keyword Editor as described in the section *Keyword Maintenance*. |
| Minimum Length | The minimum length permitted when defining a keyword. Valid values are 1 - 16 characters. The default is one character. |
| Maximum Length | The maximum length permitted when defining a keyword. Valid values are 1 - 16 characters. The default is 16 characters. |
| Dynamic Length Adjustment | The following values are permitted:<br><br>+ At runtime, each keyword must be entered in its entirety.<br>- At runtime, each keyword can be abbreviated provided that it retains uniqueness with respect to other keywords.<br>S The number of characters which must be entered for a given keyword is to be specified during keyword definition in the **ML field** of the Keyword Editor as described in the section *Keyword Maintenance*. |
| Keyword Sequence | The sequence in which keyword entries are to be processed at runtime. Possible values are 1, 2, 3 and P (for parameter indicator); the default sequence is 12, which means first the first keyword entry and then the second keyword entry. See also the **field E** as described in the section *Keyword Maintenance*. |
| Alternative Sequence | An alternative sequence in which keywords are to be processed at runtime in the event that the default sequence (specified above) results in an error during runtime. |
| Local/Global Sequence | This option specifies the order of command validation to be performed at runtime. Possible values are:<br><br>L Command is to be validated as a local command.<br>G Command is to be validated as a global command.<br><br>The default validation sequence is LG, which means that the command is to be validated first as a local command and then (if necessary) as a global one. |

### Keyword Editor Options - Header 2

Further keyword attributes can be entered on the **Processor Header Maintenance 2** screen:

| Field | Explanation |
|---|---|
| Header 1 for User Text | These two fields are used to enter a descriptive text which appears in the Keyword Editor above the column reserved for user text. This text is also output during runtime when the TEXT option is specified with the PROCESS COMMAND statement as described in the *Statements* documentation. |
| Header 2 for User Text | |
| Prefix Character 1 | This field and the next three are used to attach a hexadecimal prefix to keywords. This enables the processing of internal keywords which cannot be represented by a normal keyboard. When the command processor is cataloged, all prefix characters in keywords are replaced by the hexadecimal values specified.<br><br>If a non-blank character is entered in one of the Prefix Character fields, the specified character is replaced by the hexadecimal value specified in the Hexadecimal Replacement field. |
| Hex. Replacement 1 | The value specified in this field replaces the character specified in the field Prefix Character and is used as a prefix for a keyword at runtime. |
| Prefix Character 2 | See above Prefix Character 1. |
| Hex. Replacement 2 | See above Hex. Replacement 1. |
| Keywords in Upper Case | This option specifies whether keywords are to be translated to upper case in the Keyword Editor and the application:<br><br>Y Keywords entered in the Keyword Editor are automatically converted to upper case. In the application, end-users can enter the keywords in upper or lower case.<br><br>N Keywords entered in the Keyword Editor are not converted to upper case. In the application, end-users must enter the keywords *exactly* as they appear in the Keyword Editor. |
| Unique Keywords | This option specifies whether keywords within the processor must be unique.<br><br>Y Each keyword defined must be unique within this processor, regardless of its type.<br><br>N Each keyword defined for a given keyword type (1, 2, 3 or P) must be unique. |

### Miscellaneous Options - Header 3

Miscellaneous options can be entered on the **Processor Header Maintenance 3** screen:

| Field | Explanation |
|---|---|
| Invoke Action Editor | This option specifies whether the Runtime Action Editor is to be activated from the Function Editor (see the sections *Runtime Action Editor* and *Define Functions*).<br><br>Y The Runtime Action Editor is invoked whenever a valid keyword combination is defined in the Function Editor.<br>N The Runtime Action Editor is suppressed in the Function Editor.<br><br>**Note:** If you use the user exit NCP-REDM (delivered in the Natural system library SYSNCP), you should set this option to Y; otherwise, invalid runtime action values cannot be detected in time and can lead to runtime errors. |
| Catalog User Texts | This option specifies whether user texts are to be cataloged with the command processor.<br><br>Y Text portions of the edit line (Keyword Editor; see the section *Define Keywords*) and the user text portion of the action line (Runtime Action Editor) are bound to the associated keyword or function when the command processor is cataloged. This text can then be read at runtime using the TEXT option of the PROCESS COMMAND statement.<br>N Texts are not cataloged with the command processor and cannot be read at runtime. |
| Security Prefetch | This option specifies whether security checking is to be performed when the command processor is initially invoked during runtime or at each command evaluation.<br><br>Y If Natural Security is installed, security checking is performed for all keywords when the processor is invoked.<br>N If Natural Security is installed, security checking is performed with the evaluation of each keyword.<br><br>If option Y is selected, security checking is performed only once for all keywords when the command processor is invoked. Since the checking procedure takes time, evaluation of the first command is comparatively slow at runtime, while the evaluation of all remaining commands is comparatively fast. Conversely, if option N is selected, the evaluation time for each command is always the same because security is checked for each keyword individually before it is evaluated. |
| Command Log Size | Commands processed at runtime can be stored in a command log area by the command processor. Specify in the input field the number of KBs storage space allocated to command logging:<br><br>0 No storage space is allocated to command logging. Command logging is inactive.<br>1 1 KB of storage space is allocated to command logging. Command logging is active. |
| Implicit Keyword Entry | This option specifies whether a keyword of type 1 is to be retained as an implicit keyword for all subsequent commands. |

| Field | Explanation |
|---|---|
| | 1 If a command is entered which only contains a keyword of type 2, the command processor assumes the most recently entered keyword of type 1 as implicit keyword.<br>N Option is disabled. |
| Command Delimiter | This option specifies the character used to separate commands if more than one command is specified in the Command line. At runtime, only the first command will be executed.<br>For example:<br>DISPLAY CUSTOMER; MODIFY CUSTOMER; PRINT. |
| PF-Key may be Command | This option specifies whether commands can be allocated to PF keys: if the command processor receives at runtime a command line which contains all blanks, it checks if a PF key has been pressed by the user.<br>Possible values are:<br>A The identifier for this PF key (system variable *PF-NAME) is used as the command.<br>K The content of the *PF-KEY system variable is used as the command.<br>Y If *PF-NAME is empty, the content of the *PF-KEY system variable is used instead.<br>N PF keys cannot be used as command, Natural error NAT6913 is issued with message "Command line not accepted".<br>For more information on the system variables *PF-NAME and *PF-KEY see the *System Variables* documentation. |

### Command Data Handling - Header 4

The attributes to be entered on the **Processor Header Maintenance 4** screen specify how command data are handled for a function; command data are optional.

Options are:

| Field | Explanation |
|---|---|
| Data Delimiter | Specifies the character to be used to precede data. Default data delimiter is "#".<br>Example: ADD CUSTOMER #123 |
| Data Allowed | Specifies if data input is allowed at runtime.<br>N A runtime error occurs if data is found.<br>D Data is dropped if present.<br>S Data is placed at the top of the Natural stack. No verification is performed. |

| Field | Explanation |
|---|---|
| | Y Data is checked and keyword entries of type P (parameter indicator) are evaluated.<br><br>Example of Y: DISPLAY CUSTOMER NAME=SMITH |
| More than one Item Allowed | Only applies if the option Data Allowed is set to Y. Specifies whether more than one data string is permitted.<br><br>N A runtime error occurs if more than one data string is found.<br>D All data after the first data string are dropped.<br>Y More than one data string is permitted.<br><br>Example: ADD ARTICLE #111 #222<br><br>As long as uniqueness is guaranteed, the data delimiter can be omitted.<br><br>Example: ADD ARTICLE 123 |
| Maximum Length of one Item | Only applies if the option Data Allowed is set to Y.<br>Specifies the maximum number of characters allowed for a data string. If the specified maximum is exceeded, a runtime error occurs. Valid range: 1 - 99. |
| Item Must be Numeric | Only applies if the option Data Allowed is set to Y. Specifies whether each data value must be an integer value.<br><br>Y Data input must be a positive integer value. If not, a runtime error occurs.<br>N Data can be of any type. |
| Put to Top of Stack | Only applies if the option Data Allowed is set to Y. Specifies where data is to be placed.<br><br>Y Data is placed at the top of the Natural stack.<br>1-9 Data is placed in the *n*th occurrence of the DDM field RESULT-FIELD. If the occurrence has already been filled as a result of a runtime action, it is overwritten. |
| If Error, Drop all Data | Only applies if the option Data Allowed is set to Y or N. Specifies the reaction to a data evaluation error:<br><br>Y If an error occurs during evaluation of the data, data is discarded and processing continues.<br>N If an error occurs during data evaluation, control is given to the error handler as described below. |

### Runtime Error Handling - Header 5

The attributes to be entered on the **Processor Header Maintenance 5** screen specify how to handle runtime errors:

| Field | Explanation |
|---|---|
| General Error Program | The name of the program which is to receive control when an error is detected during runtime processing by the command processor. The Natural stack contains the following information when this program is invoked:<br><br>Error Number (N4)<br>Line Number (N4)<br>Status (A1)<br>Program Name (A8)<br>Level (N2)<br><br>If no error program and no specific error handling is specified (see below), the program with the name as contained in the Natural system variable *ERROR-TA is invoked; otherwise, a Natural system error message is issued. |
| Keyword not found | Indicates whether an action has been specified that is to be performed if a keyword could not be found. |
| Keyword missing | Indicates whether an action has been specified that is to be performed if the keyword type is missing. |
| Keyword Sequence Error | Indicates whether an action has been specified that is to be performed in the case of a keyword sequence error. |
| Command not defined | Indicates whether an action has been specified that is to be performed in the case of an undefined command. |
| Data disallowed | Indicates whether an action has been specified that is to be performed in the case of disallowed data. |
| Data Format/Length Error | Indicates whether an action has been specified that is to be performed in the case of a format/length error. |
| General Security Error | Indicates whether an action has been specified that is to be performed if an error is detected during a general security check. |
| Keyword Security Error | Indicates whether an action has been specified that is to be performed if an error is detected during a keyword security check. |
| Command Security Error | Indicates whether an action has been specified that is to be performed if an error is detected during a command security check. |

### Statistics - Header 6

The **Processor Header Maintenance 6** screen contains only output fields which report statistical data about the keywords specified for a command processor.

The following statistical information is provided:

| Field | Explanation |
|---|---|
| Entry *n* Keywords | The number of keywords of type *n* defined in the command processor (not including synonyms). |
| Entry *n* Keywords + Synonyms | The sum of keywords of type *n* and their assigned synonyms. |
| Highest IKN for Entry *n* | The largest Internal Keyword Number for the keyword of type *n*. |
| Possible Combinations | The number of possible combinations for keywords defined. |
| Cataloged Functions | The number of keyword combinations currently cataloged. |

### Status - Header 7

The **Processor Header Maintenance 7** screen contains only output fields which report the time and the date when parts of the command processor were executed or modified.

## Keyword Maintenance

Keywords are the basic components for defining functions. Before it is possible to define keywords, the header maintenance records must be created (see the section ***Header Records***).

- Define Keywords
- Editor Commands
- Positioning Commands
- Line Commands

### Define Keywords

Keywords used in commands are created with the Define Keywords function and the Keyword Editor. The Keyword Editor is similar to existing Natural editors except that lines of the editor are broken up into separate fields. Most of the **editor commands** (see the relevant section) and the **line commands** (see the relevant section) which are used in the Natural program editor can also be used in the Keyword Editor.

#### To invoke the Keyword Editor

1 In the Processor Source Maintenance menu, enter Function Code **K** (Define Keywords).

2 Press ENTER.

The Keyword Editor screen is displayed.

The Keyword Editor screen is shown below. Several keywords have already been defined to serve as examples for this section.

```
09:42:39                    - SYSNCP Keyword Editor -                 2000-05-04
Modify Keywords              Name SAGTEST   Library SYSNCP   DBID 10   FNR 32

I Line E Use  Keyword          IKN   ML Comment
- ---- - ---- ---------------- ----  -- ----------------------------------------
     1 1 Acti MENU             1004   1
     2 1 Acti DISPLAY          1002   2
     3 S Syno SHOW             1002   1
     4 1 Acti DELETE           1001   2
     5 S Syno PURGE            1001   1
     6 S Syno ERASE            1001   1
     7 1 Acti FILE             1003   4
     8 P Parm NAME             4002   2
     9 2 Obje FILE             2001   4
    10 P Parm NUMBER           4001   2
    11 2 Obje DOCUMENT         2003   2
    12 1 Acti INFORMATION      1005   1
    13
    14
- ---- - ---- ----- All ------ ----  -- ----------------------------------------

Command ===>
Enter-PF1---PF2---PF3---PF4---PF5---PF6---PF7---PF8---PF9---PF10--PF11--PF12---
      Help  Cmd   Exit  Last  List  Flip  -1    +1    Top   Bot   Info  Canc
```

Enter in the Keyword Editor all the keywords which you want to have in your command language. These can be entered in any order desired, except synonyms, which must immediately follow the keywords they are related to. To each keyword you assign a type which specifies to which part of command syntax the keyword belongs. Rules of command syntax for a command processor are specified in the processor header; see ***Keyword Runtime Options - Header 1*** in the section *Header Records*. For example, you can specify whether a keyword is to be of type 1 (entered in first position in a command), type 2, type 3, a synonym for another keyword or a parameter indicator.

**Note:** A command language requires a strict syntax because, to date, no computer is capable of understanding semantics. Word type is, therefore, the only practical way to communicate meaning in a command language.

In the example above, the keywords DELETE and DISPLAY are defined as keywords of type 1. As specified in the processor header, these keywords denote actions. The keyword DOCUMENT is defined as a keyword of type 2 and it denotes an object. The keyword FILE, however, is defined as both type 1 and 2, and it can, therefore, denote an action or an object, depending on where it is

positioned in the command. It is possible to compose the two keyword types to make commands, such as DELETE FILE and FILE DOCUMENT.

You can save the keywords you have entered by issuing the SAVE or STOW command from the Command line. In addition to saving the keyword definitions in source form, the STOW command performs a consistency check on them. Once a keyword is stowed successfully, it is given an internal keyword number (IKN) which is used at runtime to evaluate a command. Synonyms are always linked to a master keyword and always take the IKN of their master.

Each line in the Keyword Editor contains the following fields:

| Field | Explanation |
|---|---|
| I | Output field. An information field which can contain the following values:<br><br>E Indicates that a definition error has been detected.<br>X Line is marked with X.<br>Y Line is marked with Y.<br>Z Line is marked with both X and Y.<br>S Scan value found in this line. |
| Line | Output field. The line number of the editor. |
| E | Specifies the entry type for a keyword; that is, the position the keyword is to be entered in a command: first, second or third position, synonym or parameter indicator.<br><br>For instance, in the **Keyword Editor screen** example above the keyword DELETE is of entry type 1 and DOCUMENT of type 2. Using these keywords, the command DELETE DOCUMENT can be defined.<br><br>The field takes any of the following characters as input:<br><br>1 The keyword defined in this line is to be used as the first titem in a command sequence.<br>2 The keyword defined in this line is to be used as the second titem in a command sequence.<br>3 The keyword defined in this line is to be used as the third titem in a command sequence.<br>S The keyword defined in this line is to be used as a synonym for the preceding keyword with titem type 1, 2, 3 or P.<br>P The keyword defined in this line is to be used as a parameter indicator in a command sequence.<br>* No keyword is to be defined in this line. Instead, the line is to be used solely as a comment line.<br>? This symbol is an output value which indicates an invalid keyword specification. |
| Use | Output field. The value displayed is determined by the value entered in the preceding field E: |

| Field | Explanation |
|---|---|
| | 1-3 The first four characters of the user text specified in the processor header for the first, second and third keyword entries respectively are displayed. See also ***Keyword Editor Options - Header 2*** in the section *Header Records*.<br>S SYNO, the abbreviation for synonym, is displayed.<br>P PARM, the abbreviation for parameter indicator, is displayed. |
| Keyword | Enter the keyword to be defined. Embedded blanks are not permitted. If you have specified in the processor header that keywords can only be upper case, then keywords are always translated to upper case, regardless of how they are entered. Otherwise, the case remains as entered.<br><br>The maximum and minimum length of keywords depends on the settings specified in the header (default: 1 - 16 characters). Keywords must be unique unless specified otherwise in the header. Keyword prefixes can be used as described in ***Keyword Editor Options - Header 2*** in the section *Header Records*. |
| IKN | Output field. The Internal Keyword Number (IKN) is an identifier assigned to each valid keyword. IKNs are useful for testing and debugging. They are allocated only when a keyword is successfully stowed (see also the **STOW** command under *Editor Commands*). Each keyword is assigned a unique IKN, except synonyms, which take the IKN of their master term (see the **Keyword Editor screen** example above: DISPLAY and SHOW). |
| ML | Input and output field indicating the minimum length of a keyword. The field is an input field if **S** is specified in the **Dynamic Length Adjustment** field of the processor header as described in *Keyword Runtime Options - Header 1*, *Header Records*. In this case, you must specify the number of characters which must be entered for the keyword. For all other input, this field contains the minimum number of characters of a keyword a user must specify to avoid ambiguity with other keywords.<br><br>For instance, in the **Keyword Editor screen** example above, keyword MENU requires only input of **M** while keyword DISPLAY requires input of **DI** to avoid ambiguity with keyword DELETE. |
| Comment | Enter free text for a keyword. There are no input restrictions. The user text is included in the cataloged command processor if the field **Catalog User Texts** is set to Y in the header definition as described in "Miscellaneous Options - Header 3", Header Records. It can be read at runtime using the TEXT option of the PROCESS COMMAND statement. The header text appearing at the top of this column is controlled by the header definition fields "Header for User Text 1" and "Header for User Text 2". |

## Editor Commands

In the Command line of the Keyword Editor, you can enter the following commands:

| Command | Function |
|---|---|
| ADD | Adds ten empty lines to the end of the editor. |
| CANCEL | Returns to Processor Maintenance Menu. |
| CHECK | Tests the keyword source for consistency. |
| EXIT | Returns to Processor Maintenance Menu. |
| HELP | Displays valid escape characters and other useful processor settings. |
| INFO | Displays information on the keyword on which your cursor is positioned. |
| LET | Undoes all modifications made to the current screen since the last time ENTER was pressed. |
| POINT | Positions the line in which a line command **.N** is entered to the top of the current screen. |
| RECOVER | Returns keyword source that existed before last SAVE/STOW. |
| RESET | Deletes the current X and Y line markers. |
| SAVE | Keyword source is saved. |
| SCAN | Scans for the next occurrence of the scan value. |
| STOW | Keyword source is stowed and Internal Keyword Numbers (IKNs) are generated for valid keywords. |

### Positioning Commands

Editor positioning commands are the same as the ones provided for the Natural program editor. For more information, see the description of the program editor in the *Editors* documentation.

The last line of the editor contains an output field which informs you of where your display is located in the editor. The following output values are displayed:

| | |
|---|---|
| Top | Editor is currently positioned at the top of the keyword source. |
| Mid | Editor is currently positioned at the center of the keyword source. |
| Bot | Editor is currently positioned at the bottom of the keyword source. |
| Emp | Editor is currently empty. |
| All | The entire source is contained on the current screen. |

### Line Commands

Line commands in the Keyword Editor are the same as in the Natural program editor with the exception of the commands .J and .S, which cannot be used.

Each command is entered beginning in the **E** field; the remaining part of the command is entered in the **Keyword** field, as illustrated in the screen below:

```
09:42:39                    - SYSNCP Keyword Editor -                 2000-05-04
Modify Keywords              Name SAGTEST   Library SYSNCP   DBID 10   FNR 32

I Line E Use  Keyword          IKN   ML Comment
- ---- - ---- ---------------- ----  -- ----------------------------------------
     1 1 Acti MENU             1004   1
     2 1 Acti DISPLAY          1002   2
     3 S Syno SHOW             1002   1
     4 . Acti i(3)TE           1001   2
     5 S Syno PURGE            1001   1
```

**Caution:** When you move (.M) or copy (.C) lines, ensure that individual keywords are always moved or copied together with their synonyms.

When you delete (.D) lines, the corresponding keywords and any functions containing these keywords will not be deleted from the database until you issue the STOW editor command. As long as you do not issue the STOW command, these functions will still be displayed within the Function Editor.

## Function Maintenance

Functions are composed of the keywords entered in the Keyword Editor. Before it is possible to define functions, the keywords must be successfully stowed (see the section *Keyword Maintenance*).

- Define Functions
- Editor Commands
- Direct Command QUICK-EDIT
- Local and Global Functions
- Procedure for Validating Functions

### Define Functions

Use the Define Functions function and the Function Editor to specify functions and compose valid commands which can be accessed from a specific location.

#### To invoke the Function Editor

1. In the Processor Source Maintenance menu, enter Function Code **F** (Define Functions).
2. Press ENTER.

   The Function Editor screen is displayed.

The Function Editor displays all possible combinations of the keywords stowed in the Keyword Editor.

The screen below, shows the Function Editor with keywords used as examples in the **Keyword Editor screen** in the section *Keyword Maintenance*:

```
09:45:53                ***** NATURAL SYSNCP UTILITY *****            2000-05-04
User SAG                       - Function Editor -
Edit Global Combinations    Name SAGTEST  Library SYSNCP    DBID 10   FNR 32

Global
I Ac   Action           Object           Addition        Global Local Any Loc
- --   ---------------- ---------------- --------------- ------ ----- -------
       DELETE
       DELETE           DOCUMENT                                        Yes
       DELETE           FILE                                            Yes
       DISPLAY
       DISPLAY          DOCUMENT                                        Yes
       DISPLAY          FILE                                            Yes
       FILE
       FILE             DOCUMENT                                        Yes
       FILE             FILE                                            Yes
       INFORMATION                                          Yes
       INFORMATION      DOCUMENT
       INFORMATION      FILE
Repos: _______________ _______________ _______________ ------ ----- -------

Command ===>
Enter-PF1---PF2---PF3---PF4---PF5---PF6---PF7---PF8---PF9---PF10--PF11--PF12---
      Help  Cmd   Exit  Last  List  Flip        +     Top    Loc  Loc+  Canc
```

You have to validate each keyword combination that you want to designate as a valid function in your application. A keyword combination can be validated as a global function, local function or both. A global function can be invoked from anywhere in an application, whereas a local function can only be invoked from a specific location within an application.

Two fields in the upper left corner of this screen indicate the current validation mode (local or global) and the location for which keyword combinations can currently be validated. In the screen above, the text "Edit Global Combinations" indicates that global mode is active. If the local mode were active, the text "Edit Local Combinations" would appear here. In the screen above, the text "Global" appears below this text. This indicates that global validation can be performed for all of the combinations listed. In local mode, in this field the name of the location appears for which local validation can be performed (for example, "Local DISPLAY FILE").

The Function Editor contains the following columns:

| Column | Explanation |
|---|---|
| I | Output field. The following values are output as a result of function editing.<br><br>E Runtime action edited.<br>D Referenced locations displayed.<br>V Validation issued.<br>R Validation removed. |
| Ac | Action to be taken. The following values can be entered:<br><br>VG Validate as global function.<br>VL Validate as local function.<br>RG Remove validation as global function.<br>RL Remove validation as local function.<br>DL Display all functions which reference the specified function as a local function.<br>EG Invoke the Runtime Action Editor for a global function (see ***Runtime Action Editor*** in the section *Runtime Actions*).<br>EL Invoke the Runtime Action Editor for a local function (see ***Runtime Action Editor*** in the section *Runtime Actions*).<br>+G Invoke global mode, so that you can maintain any global functions.<br>+L Invoke local mode for the current line, so that you can maintain local functions for this line.<br>IN Information about keywords in this line. |
| Action | These three columns are used to display all possible combinations of currently defined keywords.<br><br>The text which appears at the top of each keyword column is controlled by the fields **First Entry used as**, **Second Entry used as** and **Third Entry used as** as specified in the processor header (see ***Keyword Runtime Options - Header 1*** in the section *Header Records*). |
| Object | |
| Addition | |
| Global | If the function has been defined as a global command, Yes appears in this field. |
| Local | If the function has been defined as a local command, Yes appears in this field for the current location (only displayed in local mode). |
| Any Loc | Any Location. If the function has been defined as a local command anywhere else within the processor, Yes appears in this field for any other location. |

### Editor Commands

In the Command line of the Function Editor, you can enter the following commands:

| Command | Function |
|---|---|
| ANY ON | Enable the column Any Loc. |
| ANY OFF | Disable the column Any Loc (the column will be filled with question marks). This allows for faster scrolling in the Function Editor. Moreover, the third repositioning field is available. Also, processing-in-progress information windows will not be displayed. |
| FIELD | Display keyword-specific combinations. |
| GLOBAL | Activate global mode. |
| LOC | Position to next location group. |
| LOC+ | Position forward by one location. |
| SINGLE ON | Display only single-word functions. |
| SINGLE OFF | Display all possible combinations. |
| TOP | Position to top of list. |

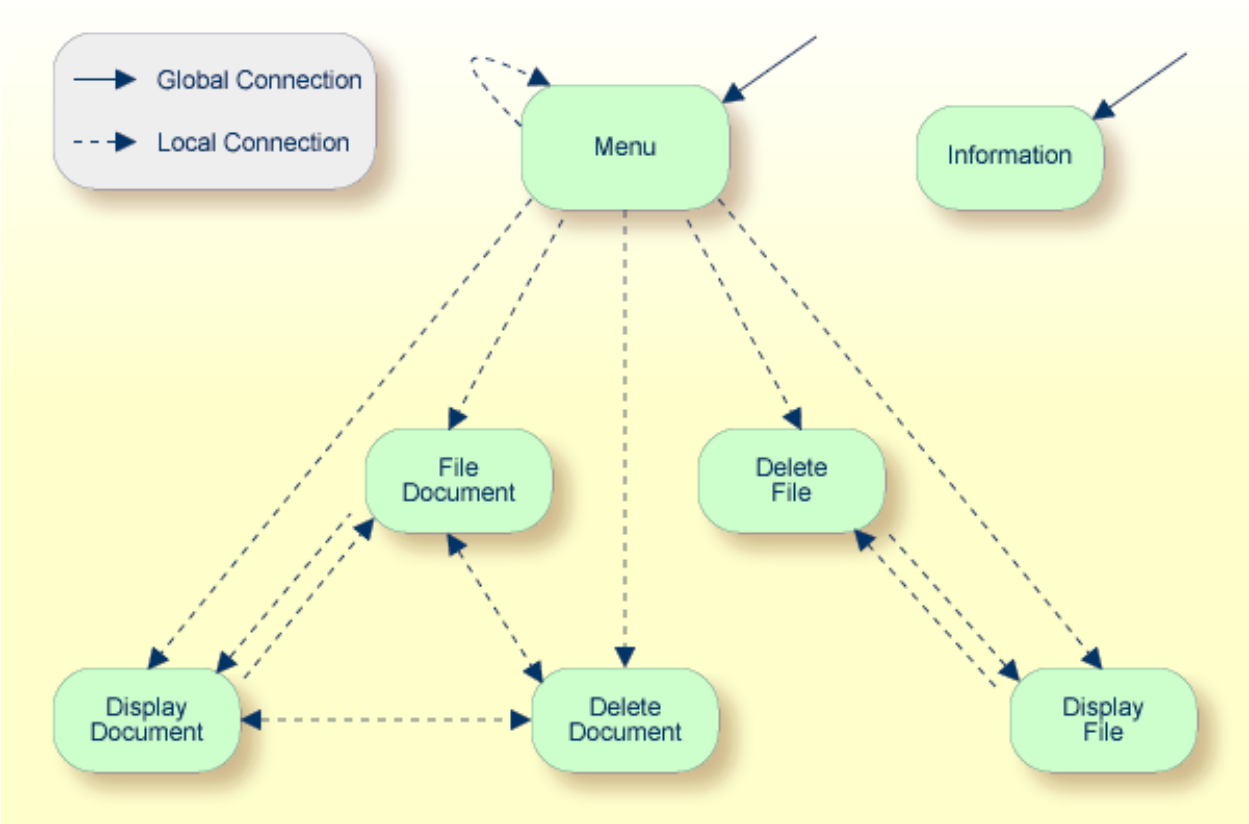
### Direct Command QUICK-EDIT

The direct command QUICK-EDIT enables you to quickly define local/global functions, as well as the corresponding runtime actions, by entering keywords or IKNs directly. This may be helpful for extremely large command processors. Note, however, that the location from which the command can be issued is not verified and navigation may not function correctly at runtime.

### Local and Global Functions

To understand the concept of local and global functions, you have to picture each valid keyword combination as a location in your application (for example, a location called Display File). In the Function Editor, you specify the commands which can be issued from this location, as well as from which locations this location can be reached using the command DISPLAY FILE.

**Local and Global Connections within a Sample Application:**




In the sample application above, the Menu and Information locations are the only locations which have been designated as global. Thus, they can be accessed directly from all of the remaining locations in the application. All locations have been designated as local to the location Menu, except Information. The only way to get from the location Display File to Display Document is via Menu.

## Procedure for Validating Functions

The Function Editor operates in two modes: global and local. From global mode you can validate global functions and from local mode you can validate global and local functions. Global mode is the default mode. You can determine whether the editor is in global or local mode by the output field above the **I** field in the editor. If the editor is in global mode, then Global is displayed. If the editor is in local mode, then the location for which local functions are to be validated is displayed. Below is a general procedure for validating global and local functions for an application.

### To validate global and local functions

1 With the Function Editor in global mode, enter **VG** (validate global) in the Ac field next to the corresponding action to validate all global functions.

Press ENTER.

The **Runtime Action Definition** screen appears.

2 Press PF3 to return to the Function Editor.

`Yes` appears under the column heading Global beside the validated functions.

3 Enter **+L** in the Ac field for each global function validated in the previous step, to switch to local mode.

Press ENTER.

4 Enter **VL** (validate local) in the **Ac** field for each function that is to serve as a location for this global function.

Press ENTER.

The Runtime Action Definition screen appears.

5 Press PF3 to return to the Function Editor.

`Yes` appears under the column heading Local beside the validated functions.

6 To validate local functions for a *local* location: Enter **+L** (invoke local mode) in the **Ac** field for each location validated in the previous step, to validate all local functions which are to be used from this location.

Press ENTER.

7 Enter **VL** (validate local) in the **Ac** field for each function that is to serve as a local function for the current location.

8 Press PF3 to return to the Function Editor.

`Yes` appears under the column heading Local beside the validated functions.

**Note:** If in the command processor header (Processor Header Maintenance 3) the field Invoke Action Editor is set to `Y`, in addition, the window Runtime Action Definition (see ***Runtime Action Editor*** in the section *Runtime Actions*) is displayed for each action.

## Runtime Actions

Once valid keyword combinations have been identified as either local or global functions in the Function Editor, it is possible to link each function with one or more runtime actions. Runtime actions consist of one or more steps which are to be carried out whenever a function is issued.

Below is information on:

- Define Runtime Actions

- Runtime Action Editor

## Define Runtime Actions

There are two different locations in SYSNCP from which you can define runtime actions: the Function Editor (see the section *Function Maintenance*) and the Result Editor. The Result Editor is explained in this section, including how to specify runtime actions for a function.

### To invoke the Result Editor

1 In the **Processor Source Maintenance** menu, enter Function Code **R** (Define Runtime Actions).

2 Press ENTER.

The Result Editor screen is displayed:

```
09:47:03              ***** NATURAL SYSNCP UTILITY *****          2000-05-04
User SAG                          - Result Editor -
List defined combinations   Name SAGTEST  Library SYSNCP   DBID 10    FNR 32


I Ac Location                          Command                          Result
- -- ------------------------------- -------------------------------- --------
     < Global >                        MENU                              KR
     < Global >                        INFORMATION                       SF
     DELETE FILE                       DISPLAY FILE                      SF
     DELETE DOCUMENT                   DISPLAY DOCUMENT                  SF
     DISPLAY FILE                      DELETE FILE                       SF
     DISPLAY DOCUMENT                  DELETE DOCUMENT                   SF
     DISPLAY DOCUMENT                  FILE DOCUMENT                     SF
     FILE DOCUMENT                     DELETE DOCUMENT                   SF
     FILE DOCUMENT                     DISPLAY DOCUMENT                  SF
     MENU                              DELETE FILE                       KCS
     MENU                              DELETE DOCUMENT                   KCCS
     MENU                              DISPLAY FILE                      KRCS
Repo ________________________________ ________________________________ --------

 Command ===>
Enter-PF1---PF2---PF3---PF4---PF5---PF6---PF7---PF8---PF9---PF10--PF11--PF12---
      Help  Cmd   Exit  Last  List  Flip       +     Top   Loc-- Loc+  Canc
```

The Result Editor contains all of the local and global functions specified in the Function Editor. Each line in the editor represents the location from which a command can be issued (Location field), the command itself (Command field) and an abbreviated summary of the action to be carried out when the command is issued (Result field).

The fields of the screen are explained in detail in the table below:

| Field | Explanation |
|---|---|
| I | Output field. Information on the last action carried out on this line. |
| Ac | Action to be taken. The following values can be entered:<br><br>DI Display the runtime action definitions for this function.<br>ED Edit the runtime action definitions for this function.<br>PU Purge this function. |
| Location | Output field.<br><br>The location within the application from which the command (see Command field below) can be issued. If the function is global, then < Global > appears in this field (the command can be issued from any location). |
| Command | Output field. The command.<br><br>The contents of the Location and Command fields may be truncated if very long keywords are used. |
| Result | Output field.<br><br>Contains an abbreviated summary of the action to be performed when the command is issued. The first character represents the **Keep Location** information (see the following section); for all other characters, see the **Runtime Action Definition** table below. |

## Runtime Action Editor

The Runtime Action Editor is used to define the actions to be taken when a command is issued from a specific location. The editor can only be invoked for functions which have been defined as global or local functions. The editor can be invoked either from the Function Editor or the Result Editor.

### To invoke the Runtime Action Editor from the Function Editor

1 In the **Ac** field, enter **EG** (edit global) for global functions.

Or:

In the **Ac** field, enter **EL** (edit local) for local functions.

2 Press ENTER.

### To invoke the Runtime Action Editor from the Result Editor

1 In the **Ac** field, enter **ED**.

2 Press ENTER.

The **Runtime Action Definition** window is displayed:

```
                        Runtime Action Definition

Location .... DISPLAY DOCUMENT
Command ..... DELETE DOCUMENT

Keep Location .... S
Data allowed ..... Y   More than one .... N   Max. Length ...... 99
Numeric .......... N   TOP of STACK ..... Y   Error: Drop ...... Y

A  Runtime Action Definition
-  ------------------------------------------------------------
F  DE-PGM_____________________________________________________
_  ______________________________________________________________
_  ______________________________________________________________
_  ______________________________________________________________
_  ______________________________________________________________
_  ______________________________________________________________
_  ______________________________________________________________
_  ______________________________________________________________
```

Actions are always associated with an origin and a destination. The origin is the location from which the command is issued, and the destination is the command itself. Thus, it is possible to link different actions to a command based on the context in which it is used.

In the Runtime Action Editor, you also specify whether the location is to remain the same after the actions have been carried out, or whether the command itself is to become the new current location.

Actions are specified by entering a single-letter code in the left column of the editor. Enter any parameters accompanying an action in the field next to the code. If the characters "/*" are entered in this field, all subsequent input is considered a comment. If you omit a required parameter, you will be prompted for input.

The sequence in which actions are performed at runtime is determined by the order of entry in the editor (from top to bottom). Thus, if a FETCH is specified, all of the actions specified below it are not to be performed.

The Runtime Action Editor contains the following fields:

| Field | Explanation |
|---|---|
| Location | Output field. The location from which the command is issued. If the function is defined as global, the field shows < Global >. |
| Command | Output field. Command for which actions are to be specified. |
| Keep Location | Specifies whether the current or a new location is to be active once the actions have been performed. A value in this field only affects commands with a specified EXEC option. Possible values are:<br><br>K Keep current location. The actions to be performed affect the current location only.<br>S Set new location (global/local). Once the actions are performed, the command processor makes the command the new current location. Every command entered subsequently has to be either a local command of this new location or a global command.<br><br>**Note:** The defined actions themselves have no influence on the location; that is, any action performed does *not* cause the current location to be changed. |
| Other Options | All other options are related to the handling of parameters provided with this command sequence. For further information, see ***Command Data Handling - Header 4*** in the section *Header Records*.<br><br>To activate the header defaults of these options, enter an asterisk (*). |

### To define runtime actions

1 Invoke the **Runtime Action Definition** window as described earlier.

2 In the field **A**, enter an action code and the corresponding action in the field opposite to it:

| Code | Runtime Action Definition |
|---|---|
| V | Default value. No runtime action is specified. |
| T | Text which can be read at runtime using the TEXT or GET option of the PROCESS COMMAND statement. |
| M | Modify command line. The data are placed in the command line. |
| C | Command. This command is placed at the top of the Natural stack. If an asterisk (*) is specified here, the name of the program which issued this PROCESS COMMAND statement is put on top of the stack (STACK TOP COMMAND '*PROGRAM'). (*) |
| D | Data. These data are placed on top of the Natural stack. (*) |
| F | Natural program name. The program is invoked with a FETCH statement. (*) |
| S | Natural STOP statement. The statement is executed at runtime. (*) |
| E | The value specified in this line is to be moved immediately into the system variable *ERROR-NR. |
| R | A return code is entered in the DDM field RETURN-CODE as described in PROCESS COMMAND in the *Statements* documentation. |
| 1 to 9 | A text string. This value is entered into the multiple DDM field RESULT-FIELD as described in PROCESS COMMAND in the *Statements* documentation. |

| Code | Runtime Action Definition |
|---|---|
| * | Comment line. |

* These actions are only performed with the EXEC option of the PROCESS COMMAND statement.

3 Press PF3 to leave the **Runtime Action Definition** window.

**Note:** The user exit NCP-REAM allows you to use some or all of the above codes. The user exit NCP-REEM allows you to modify the line that follows the heading of the Runtime Action Definition table. The user exit NCP-REDM allows you to define default values for runtime action definitions (if you use this user exit, see also ***Invoke Action Editor*** in the section *Header Records*). All user exits mentioned above are delivered in the Natural system library SYSNCP.

## Processor Cataloging

Once you have specified runtime actions for all of the functions you want to use in your command processor, you should catalog the command processor. Cataloging a command processor generates a Natural object of type Processor.

### To catalog a command processor

1 In the Processor Maintenance menu, enter Function Code **C** (Catalog Processor),
the name of the command processor to be cataloged,
and the name of the Natural library in which the command processor is to be cataloged.

2 Press ENTER.

**Note:** If you have Natural Security installed, you have to allow the use of your command processor as described in the *Natural Security* documentation in the section *Functional Security*.

## Administrator Services

SYSNCP provides facilities for the administration of command processors. Only system administrators, as defined in *Natural Security*, are authorized to access these services.

### To access the administrative services

1 In the **Processor Source Maintenance** menu, enter Function Code **A** (Administrator Services).

2 Press ENTER.

The Administrator Services screen is displayed:

```
09:49:11                ***** NATURAL SYSNCP UTILITY *****          2000-05-04
User SAG                       - Administrator Services -

                       Code   Function

                        S     Select Processor
                        C     Copy Processor Source
                        D     Delete Processor Source
                        P     Print Source/Object/NCP-Buffer
                        U     Unload Processor to Work File 3
                        L     Load Processor from Work File 3
                        F     Freeze Processor Source
                        R     References from Natural Security
                        ?     Help
                        .     Exit

                 Code .. _     Name .. SAGTEST_  Library .. SYSNCP__




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

**Note:** If you do not have Natural Security installed, be aware that all other users have administrator status.

Below is information on:

- Select Processor
- Copy Processor Source
- Delete Processor Source
- Print Source/Object/NCP Buffer
- Unload Processor
- Load Processor
- Freeze Processor Source

- References from Natural Security

### Select Processor

See the section ***Processor Selection***.

### Copy Processor Source

In copying processor sources, you have the choice of copying the entire processor or only selected sources (header, keywords, functions, runtime action definitions).

#### To copy a command processor

1 In the Administrator Services menu, enter Function Code **C**.

2 Press ENTER.

The Copy Processor Source window is displayed to provide source and target information:

```
                        Copy Processor Source

               Source           Target

Name ........ SAGTEST_         ________
Library ..... SYSNCP__         SYSNCP__
DBID ........ 10___            10___
FNR ......... 32___            32___
Password ....
Cipher Key ..

Replace ..... NO_
```

3 In the Source fields, enter the name of the processor to be copied, and the library, database ID (DBID) and file number (FNR) in which the processor is stored. The default values correspond to the processor specified on the **Administrator Services** menu.

In the **Target** fields, enter the name of the processor to be copied to, and the library, database ID (DBID) and file number (FNR) into which the processor is to be copied.

In the **Cipher Key** field, enter the appropriate password and/or cipher key if the source and/or target file is protected by a password and/or cipher key.

In the **Replace** field, enter `YES` if you want to overwrite a processor in the target environment. The default for this field is `NO`.

4 Press ENTER.

The following window is displayed to select sources:

```
                          Copy Processor Source

 Mark  Copy                            Source  Target
 ----  ----------------------------    ------  ------
  _    Header ......................    yes      no
  _    Keywords ....................    yes      no

  _    Functions ...................    yes      no
       Runtime Action Definitions ..    no       no


Source Name SAGTEST     Library SYSNCP    DBID 10    FNR 32
Target Name TEST2       Library SYSNCP    DBID 10    FNR 32

Replace ... NO
```

5 In the appropriate **Mark** fields, enter any character to select the sources you want to copy.

6 Press ENTER.

## Delete Processor Source

This function is used to delete processor sources.

### To delete a command processor

1 In the **Administrator Services** menu, enter Function Code **D**.

2 Press ENTER.

The **Delete Processor Source** window is displayed.

3 Specify the name of the processor to be deleted, and the library, database ID and file number in which the processor is stored. If the file is protected by a password and/or cipher key, you also have to enter the appropriate password and/or cipher key.

4 Press ENTER.

The following window is displayed to select the sources to be deleted:

```
                    Delete Processor Source

Mark  Delete                           Available
----  ----------------------------   ---------
 _    Header ......................    yes
 _    Keywords ....................    yes
 _    Functions ...................    yes
 _    Runtime Action Definitions ..    yes


Name SAGTEST   Library SYSNCP    DBID 10      FNR 32
```

To the right of each processor source (header, keywords, functions, runtime action definitions) is a field which indicates whether the source exists. As command processor creation is a cumulative activity, you cannot delete a source without deleting all sources which are based on it. Thus, for example, in the screen above, you cannot delete the source of the functions without also deleting the source of the runtime action definitions.

5 In the appropriate **Mark** fields, enter any character to select each source indicated as **Available**.

6 Press ENTER.

### Print Source/Object/NCP Buffer

In addition to processor sources, you can also print the processor object and the NCP.

#### To print a command processor item

1 In the **Administrator Services** menu, enter Function Code **P**.

2 Press ENTER.

The **Print Source/Object/NCP-Buffer** window is displayed.

3 Specify the name of the processor to be printed, and the library, database ID and file number in which the processor is stored. If the file is protected by a password and/or cipher key, you also have to enter the appropriate password and/or cipher key.

4 Press ENTER.

5 The following window is displayed to select items for printing:

```
                Print Source/Object/NCP-Buffer

Mark  Print                           Available
----  ----------------------------    ---------
 _    Header ......................   yes
 _    Keywords ....................   yes

 _    Functions ...................   yes
 _    Runtime Action Definitions ..   yes

 _    Processor Object ............   yes
      NCP-Buffer ..................   no

      Printer .....................   ________


Name SAGTEST   Library SYSNCP    DBID 10      FNR 32
```

To the right of each processor source (header, keywords, functions, runtime action definitions) is a field which indicates whether the item exists.

Possible input values for the **Printer** field are the logical printer ID, VIDEO or SOURCE; see also DEFINE PRINTER in the *Statements* documentation.

6 In the appropriate **Mark** fields, enter any character to select the items you want to have printed and enter the logical printer name or the value VIDEO or SOURCE in the Printer field.

7 Press ENTER.

## Unload Processor

### To unload a command processor

1 In the **Administrator Services** menu, enter Function Code **U**.

2 Press ENTER. The **Unload Processor to Work File 3** window is displayed:

```
                    Unload Processor to Work File 3

                 Source            Target

Name ........ SAGTEST_
Library ..... SYSNCP__          SYSNCP__
DBID ........ 10___
FNR ......... 32___
Password ....
Cipher Key ..

Report ...... NO_
```

3 In the **Source** fields, enter the name of the processor to be unloaded, the library, database ID, and file number in which the processor can be found; the default value is the processor specified in the **Administrator Services** menu. Enter the appropriate password and/or cipher key if the file is protected by a password and/or cipher key.

4 In the **Report** field, enter `YES` if you want a report to be produced. Default is `NO`. You do not have to use a file extension. If you wish to use an extension, you must use the file extension ".sag".

5 Press ENTER.

When the processor is unloaded, all processor sources (header, keywords, functions, runtime action definitions) are written to Work File 3.

**Note:** Use the **Object Handler** to transfer command processors from one hardware platform to another.

## Load Processor

### To load a command processor

1 In the **Administrator Services** menu, enter Function Code **L**.

2 Press ENTER.

The **Load Processor from Work File 3** window is displayed for loading processors from Work File 3 to a Natural library:

```
        Load Processor from Work File 3

  Replace existing processors .. N
  Produce load report .......... NO_
```

3 In the **Replace existing processors** field, enter **Y** or **N** (default is N) to specify whether existing processors with the same name are to be replaced by the processor to be loaded.

4 In the **Produce load report** field, enter `YES` (default is `NO`) if you want a report to be produced.

5 Press ENTER.

**Note:** Input for the processor name and the library into which the processor is to be loaded is taken from the work file.

### Freeze Processor Source

You can freeze a processor in its current state to prevent users from modifying it further.

#### To freeze a command processor

1 In the **Administrator Services** menu, enter Function Code **F**.

2 Press ENTER. The **Freeze Processor Source** window is displayed.

3 Specify the name of the processor to be frozen, and the library, database ID and file number in which the processor is stored. If the file is protected by a password and/or cipher key, you also have to enter the appropriate password and/or cipher key.

4 Press ENTER.

5 In the following window, specify with **Y** or **N** whether modification of the processor sources is to be allowed or not. Default is **Y**.

6 Press ENTER.

### References from Natural Security

This function is only available if Natural Security is active in your environment. It is used to delete functional security references from Natural Security.

If functional security is defined for a processor in Natural Security, references are created automatically. These references are stored in the FNAT/FUSER system files along with the processor sources, not in FSEC.

### To invoke References from Natural Security function

1 In the **Administrator Services** menu, enter Function Code **R**.

2 Press ENTER.

The **Delete References** window appears.

3 Specify the name of the processor, and the library, database ID and file number in which the processor is stored. If the file is protected by a password and/or cipher key, you also have to enter the appropriate password and/or cipher key.

4 Press ENTER.

5 In the following window, you can delete main references, function references and auxiliary references.

For further information on functional security for command processors, refer to the section *Functional Security* in the *Natural Security* documentation.

## Session Profile

A session profile is a collection of user-definable defaults which determine how the SYSNCP screens appear or how SYSNCP reacts to input. In a session profile, for example, you can determine which command processor you want as default for a session or which colors you want assigned to screen attributes. In SYSNCP, there is a standard session profile called STANDARD which is issued to all new users. You can create several different session profiles and activate them as required.

Administrators for SYSNCP can access and modify any session profile in SYSNCP. Other users can access all session profiles, but can modify only those session profiles which are created under their user ID or which have the same name as their user ID.

### To define or modify a session profile

- Issue the `PROFILE` command from the Command line of the **Processor Source Maintenance** menu.

  The first of three session profile maintenance screens is displayed.

Below is information on:

- Session Profile Name
- Session Parameters - Profile 1
- Color Attributes - Profile 2

- Miscellaneous Attributes - Profile 3

### Session Profile Name

The standard profile STANDARD or the value of the system variable *USER is taken as default for the profile name.

If you are defining a new session profile, the parameters/attributes are defaults. You can modify these defaults as required and save them by entering the new name and pressing PF5.

The field Session Profile Name on each profile screen is both an input and output field. Thus, it is possible to define, read or save another profile from any of these screens by entering its name in the Profile Name field and pressing PF5 or PF4, respectively.

### Session Parameters - Profile 1

On the first profile maintenance screen, you can modify the following fields:

| Field | Explanation |
|---|---|
| Apply Terminal Control 1 | These fields can be used to enter the parameters of a SET CONTROL statement to be issued by SYSNCP at startup.<br><br>For example, when you enter **Z** in any of the fields, SYSNCP issues the statement `SET CONTROL 'Z'`. |
| Apply Terminal Control 2 | |
| Default Processor Name | The default command processor name to be used for this session. |
| Default Processor Library | The Natural library to be used to store a command processor. |
| Cancel Reaction | Specifies whether a warning is to be issued whenever the requested modification is not completed and the CANCEL command is issued.<br><br>W Issue warning.<br>B Back out and cancel without issuing warning. |
| Clear Key Allowed | Specifies whether clear key is allowed.<br><br>N Clear key disallowed.<br>Y Clear key active and has same effect as CANCEL. |
| Default Cursor Position | Specifies placement of the cursor.<br><br>1 Cursor to be positioned in first field of the screen.<br>C Cursor to be positioned in command line. |
| Exec/Display Last Command | Specifies action to be taken as a result of the LAST command:<br><br>E Execute last command issued in command line.<br>D Display last command issued in command line. |

### Color Attributes - Profile 2

On the second profile maintenance screen, you can assign colors to various screen attributes, or overwrite existing color assignments.

By specifying the following color codes, you can assign the following colors:

| Code | Color |
|---|---|
| BL | Blue |
| GR | Green |
| NE | Neutral |
| PI | Pink |
| RE | Red |
| TU | Turquoise |
| YE | Yellow |

For color assignments to screen attributes, see also the terminal command %= in the *Terminal Commands* documentation.

### Miscellaneous Attributes - Profile 3

The following attributes can be specified on the third profile maintenance screen:

| Field | Explanation |
|---|---|
| Message Line Position | The line on which messages are to be displayed. The value 21 is recommended. See also the terminal command %M in the *Terminal Commands* documentation for more information. |
| Text for PF5 Key | The PF5 function key is reserved for global (session-wide) use. The text to be displayed on the PF-key line for PF5 can be entered in this field. |
| Command for PF5 Key | The PF5 function key is reserved for global (session-wide) use. The command to be executed when PF5 is pressed can be entered in this field. |

In addition, the screen displays when and by which user this profile was last modified.

# XIX SYSPARM Utility

# 99 SYSPARM Utility


- Invoking SYSPARM ..... 730
- List Profiles ..... 731
- Display Profile ..... 733
- Add New Profile ..... 734
- Modify Profile ..... 734
- Editing Profiles ..... 735
- Copy Profile ..... 739
- Delete Profile ..... 739
- Direct Commands and Batch Processing ..... 740
- Maintaining Profiles in Different Environments ..... 748
- Invoking Help on Parameters from the Command Line ..... 748

The SYSPARM utility is used to create and maintain a set of Natural profile parameters as an individual parameter profile that can be used for each Natural session.

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

For descriptions of the individual profile parameters that can be defined in a parameter profile, refer to the *Parameter Reference* documentation.

The parameter profiles are stored under user-defined names in the specified FNAT or FUSER Natural system file.

You can restrict the use of a parameter profile to specific users by setting the profile parameter `USER` as described in the *Parameter Reference* documentation.

**Related Topic:**

- `PROFILE` in the *Parameter Reference* documentation

## Invoking SYSPARM

### To invoke the SYSPARM utility

- Issue the following Natural system command:

```
SYSPARM
```

The **Menu** of the SYSPARM utility is displayed which provides the following functions and fields:

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

| Field/Function | Explanation |
|---|---|
| **Profile** | Selects the specified parameter profile. Enter a valid profile name or use asterisk (*) notation to select a range of profiles. Use asterisk (*) or leave this field blank to select all profiles available. |
| **Copy to** | Copies the specified parameter profile. |
| **DBID** | Selects the database ID (DBID) of the Natural system file where the parameter profile is stored. Default is the current FPROF (if set), or current FNAT otherwise. |
| **FNR** | Selects the file number (FNR) of the Natural system file where the parameter profile is stored. Default is the current FPROF (if set), or current FNAT otherwise. |
| **Password** | Specifies the password (8 characters) of the Adabas file where the parameter profile is stored. |
| **Cipher** | Specifies the cipher code (8 digits) of the Adabas file where the parameter profile is stored. |

The SYSPARM functions listed above are described in the remainder of this documentation.

## List Profiles

This function is used to list all parameter profiles contained in the specified Natural system file. From the parameter profile list, you can view, modify or delete one or more profiles. You can also display a property such as the code page of a parameter profile or when and by whom it has been modified at last.

### To list parameter profiles

- In the SYSPARM **Menu**, enter function code `L` and, in the **Profile** field, enter the name of a parameter profile or specify a range of names:
  - Enter an asterisk (*) or leave the field blank to list all parameter profiles.
  - Use asterisk (*) notation to list all parameter profiles with names that start with a specified value, where value is any combination of one or more characters, for example: `ABC*`
  - Use the greater than (>) sign to list all parameter profiles with names greater than or equal to a specified value, for example: `ABC>`
  - Use the less than (<) sign to list all parameter profiles with names less than a specified value, for example: `ABC<`.

  The **List Profiles** screen is displayed with a list of all parameter profiles of the specified name range.

  You can press PF7 and PF8 to scroll up or down one page in the list. You can also press PF11 (>/<) to toggle properties of the profile, such as when (**Date**, **Time**) and by whom (**User ID**)

the profile has been modified at last, the Natural version (**Version**) that was current when the file has been modified and the code page of the profile (**Code Page**).

**Note:** If an entry for **Code Page** contains more than 17 characters, it is truncated. You can recognize a truncated entry from the trailing greater than sign (>). By pressing PF11 (>) once more, the code page's full name is displayed.

### To view, modify or delete a single parameter profile

- Place the cursor anywhere in the **Sel.** or **Profile** column of the parameter profile you want to process and press one of the following PF keys:

| PF Key | Line Command | Function |
|---|---|---|
| PF4 | D | Invokes the **Display Profile** function. |
| PF5 | M | Invokes the **Modify Profile** function. |
| PF6 | X | Invokes the **Delete Profile** function. |

Or:

In the **Sel.** column, next to the parameter profile you want to process, enter one of the line commands listed above and press ENTER.

Depending on the key pressed or line command entered, the selected parameter profile is either displayed on the screen or a delete confirmation window opens for this profile.

### To view, modify or delete multiple parameter profiles

1. In the **Sel.** column, next to the parameter profiles you want to process, enter one of the line commands listed in the **table** above.

   You can use PF7 or PF8 to scroll up or down in the list of parameter profiles to go to the required items.

2. Press ENTER when you have finished entering all line commands for all parameter profiles.

   Depending on the line command entered, the first parameter profile selected is either displayed on the screen or a delete confirmation window opens for this profile.

3. When finished with the first parameter profile, press PF3 or PF12 to process the next parameter profile. PF3 also saves the current parameter profile, if modified.

## Display Profile

This function is used to display a parameter profile.

### To display a parameter profile

- In the SYSPARM **Menu**, enter function code `D` and the name of an existing parameter profile.

  Or:

  From the **List Profiles** screen, select one or more parameter profiles as described in the relevant section.

  A **Display Profile** screen similar to the example below appears with the parameter definitions of the specified parameter profile:

```
14:30:54             ***** NATURAL SYSPARM UTILITY *****            2009-02-11
                     - Display Profile TESTPROF (10,2410) -
 >  /* This is a test profile. */                                            <
 >  AUTO=ON FNAT = (102,110,PASSWORD) FUSER=(1099,1100,PASSWORD,12345678)    <
 >  RPC=(RPCSIZE=80,SRVNAME=MYSERV,SERVER=ON,DFS=(SRV2,NODE1,,ACI))          <
 >  PRINT=((2,12,18),AM=STD,DEST='PRINT**',OPEN=INITOBJ,CLOSE=CMD)           <
 >  PRINT=((1,3,6-11,15),AM=NAF)                                             <
 >  ESIZE=90                                                                 <
 >                                                                           <
 >                                                                           <
 >                                                                           <
 >                                                                           <
 >                                                                           <
 >                                                                           <
 >                                                                           <
 >                                                                           <
 >                                                                           <
 >                                                                           <
 >                                                                           <
 >                                                                           <

Command ===>









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

The screen title indicates the name of the parameter profile (here: `TESTPROF`) and the database ID and file number (here: `10,2410`) where the parameter profile is stored.

## Add New Profile

This function is used to create a new parameter profile.

### To create a parameter profile

- In the SYSPARM **Menu**, enter function code `A` and the name of the parameter profile you want to create, and press ENTER.

  A blank **edit screen** appears. For instructions on entering the parameters you want to specify in the new parameter profile, see *Editing Profiles*.

## Modify Profile

This function is used to change the parameter specifications of a parameter profile.

### To modify a parameter profile

- In the SYSPARM **Menu**, enter function code `M` and the name of an existing parameter profile, and press ENTER.

  Or:

  Invoke the **Display Profile** screen for an existing parameter profile and press PF5.

  Or:

  Invoke the **List Profiles** screen and select one or more parameter profiles as described in the relevant section.

  An **edit screen** with the parameter definitions of the specified parameter profile appears. For instructions on using this screen to add or modify definitions, see *Editing Profiles*.

## Editing Profiles

The edit screen of the SYSPARM utility appears when executing the **Add New Profile** or the **Modify Profile** function. This screen is used to enter the parameter specifications you want to include in a parameter profile. For detailed information about the profile parameters available, refer to the *Parameter Reference* documentation or use the `HELP *` command, see ***Invoking Help on Parameters from the Command Line***.

The edit screen looks similar to the example below:

```
15:13:44             ***** NATURAL SYSPARM UTILITY *****            2009-02-11
 >  /* This is a test profile. */                                            <
 >  AUTO=ON FNAT = (102,110,PASSWORD) FUSER=(1099,1100,PASSWORD,12345678)     <
 >  RPC=(RPCSIZE=80,SRVNAME=MYSERV,SERVER=ON,DFS=(SRV2,NODE1,,ACI))           <
 >  PRINT=((2,12,18),AM=STD,DEST='PRINT**',OPEN=INITOBJ,CLOSE=CMD)            <
 >  PRINT=((1,3,6-11,15),AM=NAF)                                              <
 >  ESIZE=90                                                                  <
 >                                                                            <
 >                                                                            <
 >                                                                            <
 >                                                                            <
 >                                                                            <
 >                                                                            <
 >                                                                            <
 >                                                                            <
 >                                                                            <
 >                                                                            <
 >                                                                            <
 >                                                                            <
Help with parameters .. _______                        (Profile name: TESTPROF)

Command ===>









Enter-PF1---PF2---PF3---PF4---PF5---PF6---PF7---PF8---PF9---PF10--PF11--PF12---
      Help        Exit  Check Save                    Insrt Del   Copy  Canc
```

The edit screen contains 18 input lines (marked with >) with a maximum length of 72 characters each. The individual parameters entered must be separated from one another by (one or more) blanks or commas. You can spread the specifications of the parameters over as many lines as you like.

You can enter a commentary text in each input line, or spread a comment over as many lines as you like. Comments must be preceded by `/*` and followed by `*/` as indicated in the example above.

This section covers the following topics:

- Using the PROFILE Profile Parameter
- PF Keys
- Help with Parameters

### Using the PROFILE Profile Parameter

If you need additional space for editing or want to combine different parameter profiles or group parameter profiles by categories, concatenate multiple parameter profiles by entering the `PROFILE` profile parameter as the last entry in the profile(s) concerned. The first parameter profile then evaluates the parameter strings specified in the second parameter profile as part of the string of the first parameter profile.

For example:

`PROFILE=P2` entered at the end in parameter profile P1, and `PROFILE=P3` entered at the end in parameter profile P2, firstly invokes P1 and then P2 followed by P3.

For detailed information on the `PROFILE` parameter, see the relevant section in the *Parameter Reference* documentation.

### PF Keys

The following PF keys are available on the edit screen:

| PF Key | Function |
|---|---|
| PF4 | Checks whether the parameter specifications within the parameter profile are syntactically correct. |
| PF5 | Stores the parameter profile. |
| PF9 | Inserts one blank line below the line containing the cursor. |
| PF10 | Deletes the line containing the cursor. |
| PF11 | Copies the line containing the cursor. |

### Help with Parameters

The **Help with parameters** field on the edit screen can be used to obtain help information on valid parameter specifications and incorporate new specifications into the current parameter profile.

#### To view online help and include new parameters

1 In the **Help with parameters** field, enter one of the following:

- The name of the required parameter.
- One or more characters with asterisk (*) notation (for example, `PR*`) for all parameters whose names start with the specified character(s).
- An asterisk (*) for all parameters available.

2 Do *not* press ENTER yet if you want to *insert* a new parameter specification into an existing parameter profile and proceed with the following step.

Mark the input line below which you want to insert the new parameter specification(s) by placing the cursor on this line, and then press ENTER.

Or:

Press ENTER if you want to *append* a new parameter specification to the end of the parameter profile.

3 After performing the previous step, depending on the parameter value entered earlier, one of the following appears:

- If you entered a full name, a help screen (see below) appears.
- If you specified a name range using the asterisk (*), a window opens with a list of all parameters of the specified range.

  From this list, select the required parameter by entering any character in the input field next to the required parameter and pressing ENTER.

The help screen looks similar to the example below:

```
13:08:21             ***** NATURAL SYSPARM UTILITY *****           2009-02-11

  The parameter AUTO is used to cause an automatic logon at the start
  of the Natural session.

  AUTO=ON     An automatic logon is executed at the start of the Natural
              session. The value contained in the Natural system variable
              *INIT-USER is used as the user ID for the logon.
  AUTO=OFF    No automatic logon is performed.





   Enter your parameter specification:                More Help:   (+,-)
 >
 >
 >
 >
 >

Command==>





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

The upper screen section contains help text on the specified parameter (in the example above, `AUTO`). If the text contains more than one page, you can enter a plus (+) sign before the **More Help** field to display the next page. Enter a minus (-) sign to return to the previous screen or enter a period (.) to terminate the help function and return to the edit screen.

The lower section contains five input lines (marked with >).

4 Press ENTER.

5 In the input lines, enter the required parameter specifications.

You can press PF4 to check the specifications.

6 Press PF3 to terminate the help function.

The edit screen appears with the new parameter specifications either appended to the end of the parameter profile or inserted below the input line marked with the cursor in Step 3.

**Note:** See *Invoking Help on Parameters from the Command Line* **for invoking help on parameters and subparameters from the command line.**

## Copy Profile

This function is used to copy parameter profiles.

### To copy a parameter profile

1 In the SYSPARM **Menu**, enter the following:

- Function code `C`.
- In the **Profile** field, the name of the parameter profile from which you want to copy the new parameter profile.
- In the **Copy to** field, the name of the new parameter profile.

2 Press ENTER.

A message appears indicating successful completion of the copy operation.

## Delete Profile

This function is used to delete an existing parameter profile.

### To delete a parameter profile

1 In the SYSPARM **Menu**, enter function code `X` and the name of the parameter profile you want to delete, and press ENTER.

Or:

Invoke the **List Profiles** screen and select one or more parameter profiles as described in the relevant section.

A **Delete a Profile** window opens with the name of the parameter profile you want to delete.

2 Confirm the deletion by entering the name of the parameter profile in the input field and pressing ENTER.

You can cancel a delete operation by pressing PF3 or leaving the input field blank and pressing ENTER.

A message appears indicating either successful completion or cancellation of the delete operation.

## Direct Commands and Batch Processing

The SYSPARM utility functions described earlier can also be executed by using corresponding SYSPARM commands in batch or online mode.

In addition to the functions provided on the SYSPARM utility screens, in batch mode, options are provided for specifying commentary text, the input delimiter and the input assign character.

The symbols used in the syntax diagrams shown in this section are explained in *System Command Syntax* in the *System Commands* documentation.

This section covers the following topics:

- Basic Command Syntax
- where-clause
- with-clause
- Batch Processing
- Example of SYSPARM in Batch

### Basic Command Syntax

Displayed below is the basic syntax that applies when processing SYSPARM commands.

```
                 ( { ADD     }
                   { MODIFY  }
                   { DISPLAY }  profile-name [where-clause] [with-clause]
                   { LIST    }
                 ( { DELETE  }
                   { X       }  profile-name [where-clause]
                 (   COPY       profile-name1 TO profile-name2 [where-clause]
                 ( { ASSIGN  }
                   { DELIMITER }  [character]
                 (   COMMENT  { ON  }   [comment]
                              { OFF }
                 (   TERM1ST  { ON  }   [WITH
                              { OFF }    return-code]
[SYSPARM]                                                                    ) ...
```

The keywords, the variable values to be supplied with the keywords, and the optional clauses indicated in the basic command syntax of SYSPARM are described in the following table:

| Keyword | Explanation |
|---|---|
| SYSPARM | Invokes the SYSPARM utility.<br><br>This keyword is only required when executing SYSPARM commands from the NEXT or MORE prompt. |
| ADD | Executes the **Add New Profile** function.<br><br>To terminate an ADD command, enter a period (.) in a separate line. |
| MODIFY | Executes the **Modify Profile** function.<br><br>To terminate a MODIFY command, enter a period (.) in a separate line.<br><br>To insert a commentary text into a parameter profile, place the comment between the characters /* and */.<br><br>See also ***Example Input***. |
| DISPLAY | Executes the **Display Profile** function. |
| LIST | Executes the **List Profiles** function. |
| DELETE<br><br>or<br>X | Executes the **Delete Profile** function. |
| COPY | Executes the **Copy Profile** function. |
| ASSIGN | Specifies or displays the input assign character. You need to change the input assign character before you start processing data that contains the input assign character defined for your current session. Otherwise, you will receive an appropriate Natural system error.<br><br>If no value is specified, the output contains the character defined for your current session. |
| DELIMITER | Specifies or displays the input delimiter character. You need to change the input delimiter character before you start processing data that contains the input delimiter character defined for your current session. Otherwise, you will receive an appropriate Natural system error.<br><br>If no value is specified, the output contains the character defined for your current session. |
| COMMENT | Not applicable to the command REPLACE.<br><br>Activates (ON) the comment option and writes a comment after each parameter that is modified, or deactivates (OFF) the comment option.<br><br>If activated, and if no individual *comment* (see below) has been specified, the comment inserted for each parameter contains the following data:<br><br>■ the ID of the user who last modified the parameter profile as generated by the Natural system variable *USER and<br>■ the date and time when the parameter profile was last modified as generated by the Natural system variables *DATV and *TIMX. (See also the relevant sections in the *System Variables* documentation.) |

| Keyword | Explanation | |
|---|---|---|
| | To modify a comment, use the REPLACE command (see the *with-clause*). | |
| *character* | Any special character: see the Natural session parameters ID (Input Delimiter Character) and IA (Input Assign Character) described in the *Parameter Reference* documentation. | |
| *comment* | A commentary text that is to be placed between the characters /* and */. | |
| *profile-name* | The name of a parameter profile or a range of profiles. For a range, you can specify one of the following values where *value* is any combination of one or more characters: | |
| | *value** | Selects all parameter profiles with names that start with *value*, for example: AB* selects AB and AB1 but not AA1. |
| | *value>* | Selects all parameter profiles with names greater than or equal to *value*, for example: AB> selects AB and AB1 but not AA1. |
| | *value<* | All items with names less than or equal to *value*, for example: AX< selects AB and AWW but not AXA. |
| | Name ranges are only allowed with the command LIST or, in batch mode, the commands LIST and DISPLAY. | |
| *profile-name1*<br>*profile-name2* | Only applies to the COPY command.<br><br>The source parameter profile (*profile-name1)* from which to create a new parameter profile and the new target parameter profile (*profile-name2*) into which to copy the data. | |
| *where-clause* | Indicates a ***where-clause*** described in the relevant section. | |
| *with-clause* | Indicates a ***with-clause*** described in the relevant section. | |
| TERM1ST | Only applies in batch mode.<br><br>Determines whether the Natural session terminates when a SYSPARM error or warning occurs. | |
| | OFF | The session continues. |
| | ON | The session terminates.<br><br>This is the default. |
| | WITH *return-code* | Specifies the return code to be used when SYSPARM terminates Natural.<br><br>The valid code range for *return-code* is 0 to 255.<br><br>If the WITH *return-code* clause is omitted, a return code of 4 is used by default. |
| | If TERM1ST is set to OFF and more than one warning is encountered, SYSPARM adds a value of 127 to the return code specified in the WITH *return-code* clause, up to the maximum of 255. For example: A value of 4 (default) in the WITH clause then results in a final return code of 131. | |

| Keyword | Explanation |
|---|---|
| . . . | Only applies in batch mode.<br><br>Illustrates that you can specify more than one SYSPARM function by placing each function in a separate line. |

### where-clause

The *where-clause* is optional and applies to the commands ADD, MODIFY, DISPLAY, LIST, DELETE and COPY. Its syntax is as follows:

```
[WHERE] [DBID dbid] [FNR fnr] [PASSWORD password] [CIPHER cipher]
```

The keywords of the *where-clause* and the variable value that must be supplied with each keyword are explained in the following table:

| Keyword | Valid Values |
|---|---|
| WHERE | Optional keyword that indicates the start of a *where-clause*. |
| DBID | The database ID (*dbid*) of the Adabas file where the parameter profile is stored. |
| FNR | The file number (*fnr*) of the Adabas file where the parameter profile is stored. |
| PASSWORD | The 8-character Adabas password (*password*) of the Adabas file where the parameter profile is stored. |
| CIPHER | The 8-digit cipher code (*cipher*) of the Adabas file where the parameter profile is stored. |

### with-clause

The *with-clause* is optional and applies to the commands ADD, MODIFY, DISPLAY and LIST. Its syntax is as follows:

```
[WITH] { { SCAN   } text { CODEPAGE }                                  }
       { { NOSCAN }      { DIRINFO }                                  }
       { { ASSIGN    } [character]                                    }
       { { DELIMITER }                                                }
       { COMMENT { ON  } [comment]                                    }
       { COMMENT { OFF }                                              }
       { FIRST { parm-name                    }                       }
       { FIRST { [parm-value]                 }                       }
       { FIRST { comment                      }                       }
       { LINE  parm-name=parm-value [parm-name=parm-value] ...        }
       { NEW   parm-name parm-value                                   }
       { REMOVE parm-name                                             }
       { REPLACE { ;text;text;  }                                     }
       { REPLACE { text text    }                                     }
       { UPDATE parm-name { parm-value                                }
       { UPDATE parm-name { { ADD    }                                }
       { UPDATE parm-name { { SUBTRACT }                              }
       { UPDATE parm-name { { MAX    } number [ { ADD    }           }
       { UPDATE parm-name { { MIN    }          { SUBTRACT } number ] }
       { UPDATE parm-name { { MAX    }          { MAX    }           }
       { UPDATE parm-name { { MIN    }          { MIN    }           }
       { parm-name=parm-value [parm-name=parm-value]...               }
```

**Note:** In the syntax diagram above, the following only applies to the `ADD` command: `[WITH] parm-name=parm-value [parm-name=parm-value]...`

The keywords of the *with-clause* and the variable value that must be supplied with each keyword are explained in the following table:

| Keyword | Valid Values |
|---|---|
| `WITH` | Optional keyword that indicates the start of a *with-clause*. Exception: When using the `ADD` command, you must use the keyword `WITH` to clearly separate the command from the contents of the parameter profile. |
| `SCAN` | Only applies to the commands `LIST` and `DISPLAY`.<br><br>Scans one or more parameter profiles available in the specified Natural system file for a **text** string (for example, a parameter or a parameter value) and provides a list of the parameter profiles that contain the specified **text** string. |
| `NOSCAN` | Only applies to the commands `LIST` and `DISPLAY`.<br><br>Exempts a **text** string from the scan over one or more parameter profiles available in the current Natural system file and provides a list of the parameter profiles that do *not* contain the specified text string. |

| Keyword | Valid Values |
|---|---|
| CODEPAGE | Only applies to the commands LIST.<br><br>The List Profiles screen includes the column **Code Page**. |
| DIRINFO | Only applies to the commands LIST.<br><br>The List Profiles screen includes the columns **UserID**, **Date**, **Time**, **Version**, and **Code Page**.<br><br>**Note:** Code page names longer than 17 characters are truncated, see **note above**. |
| ASSIGN | See ASSIGN in the basic command syntax. |
| DELIMITER | See DELIMITER in the basic command syntax. |
| COMMENT | See COMMENT in the basic command syntax. |
| FIRST | Moves a parameter and its value or a comment to the first position in a parameter profile. This can be required, for example, if you want to specify the profile parameter PARM.<br><br>Comments contained in the first position of a parameter profile will remain in this position. A parameter and its value will only be moved to the first position if no comment occupies this position. Otherwise, the parameter will be appended after the comment(s). A comment inserted with FIRST will always be placed in the first position, before any comment(s) that originally occupied this position. |
| LINE | Places the specified parameter(s) after LINE in a new line at the end of a parameter profile. |
| NEW | Appends a parameter to the end of a parameter profile but not necessarily in a new line. To place a parameter in a new line, use the parameter LINE. |
| REMOVE | Removes a parameter and the value assigned (including parentheses) from a parameter profile. The parameter to be removed can be a subparameter, such as the profile parameter DFS, which is a subparameter of the profile parameter RPC. |
| REPLACE | Replaces an old **text** string with a new text string contained in a parameter profile. See also the option COMMENT in the basic command syntax. |
| UPDATE | Updates the value assigned to a parameter.<br><br>If the specified parameter does not yet exist, it will be appended to the parameter profile. |
| ADD | Increases the size of a parameter value by adding the specified number. The addition can be limited to a maximum or minimum value by specifying MAX or MIN (see below). |
| SUBTRACT | Decreases the size of a parameter value by subtracting the specified number. The subtraction can be limited to a maximum or minimum value by specifying MAX or MIN (see below). |
| MAX | Specifies the maximum of a parameter value. |
| MIN | Specifies the minimum of a parameter value. |
| *text* | Any text string contained in a parameter profile.<br><br>The following applies to the commands SCAN and NOSCAN: The text string must *not* contain any blank characters.<br><br>The following applies to the REPLACE command: |

| Keyword | Valid Values |
|---|---|
| | If neither the old text nor the new text contains blank characters, place a blank character between old and new text. If the old text or the new text contains blank characters, place the input delimiter character specified between old text and new text and around the entire old/new text string.<br><br>For example:<br><br>`;This comment is old;This comment is new;`<br><br>As an alternative to the specified delimiter character, you can choose any of the following characters:<br><br>' an apostrophe<br>, a comma<br>. a period<br>; a semicolon<br>/ a slash<br>\ a back slash<br>\| a vertical bar |
| *character* | See *character* in the basic command syntax. |
| *parm-name* | The full name of a parameter. |
| *parm-value* | The value assigned to a parameter. |
| *comment* | See *comment* in the basic command syntax. |
| *number* | A numeric value. |

## Batch Processing

When processing SYSPARM in batch mode, consider the following:

- To terminate SYSPARM, in a separate line, enter a period (.) or `FIN`, where `FIN` ends the Natural session.
- To execute more than one SYSPARM function, specify each function in a separate line.
- To extend a function over two lines, enter the character defined with the session parameter CF (default is `%`) anywhere in the first line. This indicates continuation on the next line.

See also ***Example of SYSPARM in Batch***.

**Related Topics:**

*Natural in Batch Mode* (*Operations* documentation)
*Using the INPUT Statement in Non-Screen Modes* (*Statements* documentation)
*Using the INPUT Statement in Batch Mode* (*Statements* documentation)

## Example of SYSPARM in Batch

The examples shown in this section demonstrate the use of SYSPARM commands and the result of parameter profile modifications executed in batch:

- Example Profile TESTPROF - Before Job Submission
- Example Input
- Example Parameter Profile TESTPROF - After Job Execution
- Example Parameter Profile TESTPRO1 - After Job Execution

### Example Profile TESTPROF - Before Job Submission

```
/* This is a test profile. */
AUTO=ON FNAT = (102,110,PASSWORD) FUSER=(1099,1100,PASSWORD,12345678)
RPC=(RPCSIZE=80,SRVNAME=MYSERV,SERVER=ON,DFS=(SRV2,NODE1,,ACI))
PRINT=((2,12,18),AM=STD,DEST='PRINT**',OPEN=INITOBJ,CLOSE=CMD)
PRINT=((1,3,6-11,15),AM=NAF)
ESIZE=90
```

**Note:** If your current input assign or delimiter character is part of the input, you need to change this input character (see `ASSIGN` or `DELIMITER` in *Direct Commands and Batch Processing*). This can be done by adding an appropriate `ASSIGN` or `DELIMITER` assignment before the statement that would otherwise cause a runtime error. See the ***Example Input*** below.

### Example Input

```
/*JCL
.
.
.
*/
SYSPARM
COPY TEST1 TO TESTPROF WHERE DBID 10 FNR 32
DISP   TESTPROF
COMMENT ON /* TESTUSER 29.Jan.2009 */
MODIFY TESTPROF with REM DFS
   REPLACE ;test profile;test profile for SYSPARM in batch;
   NEW BPSIZE 4096
   UPDATE ESIZE  ADD  20  MAX 100
   FIRST PARM INHOUSE
   DELIMITER $
   UPDATE FUSER (,6)
   ASSIGN :
   LINE IM=D, INTENS=1, AUTO=T, MT=0, MADIO=0
   .
DISP   TESTPROF
ADD TESTPRO1 WITH /* 106,210 */
    FNAT=(106,210,PASSWORD),FUSER=(,211)
    .
```

```
DISPLAY TESTPRO1
DELETE  TESTPRO2 WHERE DBID 10 FNR 32
X TESTPRO3   WHERE DBID 10 FNR 32
.
FIN
```

**Example Parameter Profile TESTPROF - After Job Execution**

```
/* This is a test profile for SYSPARM in batch. */
PARM=INHOUSE /* TESTUSER 29.Jan.2009 */
AUTO=ON FNAT = (102,110,PASSWORD) FUSER=(,6) /* TESTUSER 29.Jan.2009 */
RPC=(RPCSIZE=80,SRVNAME=MYSERV,SERVER=ON, )
PRINT=((2,12,18),AM=STD,DEST='PRINT**',OPEN=INITOBJ,CLOSE=CMD)
PRINT=((1,3,6-11,15),AM=NAF)
ESIZE=100 /* TESTUSER 29.Jan.2009 */ BPSIZE=4096 /* TESTUSER 29.Jan.2009
 */
IM=D, INTENS=1, AUTO=T, MT=0, MADIO=0 /* TESTUSER 29.Jan.2009 */
```

**Example Parameter Profile TESTPRO1 - After Job Execution**

```
/* 106,210 */
FNAT=(106,210,PASSWORD),FUSER=(,211)
```

## Maintaining Profiles in Different Environments

The SYSPARM utility is used to maintain parameter profiles within the same FNAT or FUSER system file.

To transfer parameter profiles (for example, copy or move) between different FNAT and/or FUSER system files, and to perform a parameter profile operation (for example, delete or find) in a different environment, you can use the SYSMAIN utility. For details, see *Processing Profiles* in the *SYSMAIN Utility* documentation.

## Invoking Help on Parameters from the Command Line

With the system command `HELP` you can get online help on a parameter or a subparameter. The following commands are available:

| | |
|---|---|
| `HELP <name>` | Display the help map for parameter *name*. If there is no such parameter, search is extended to subparameters. |
| `HELP SUB <name>` | Search for subparameters only. This command is useful if *name* is ambiguous, that is *name* can refer to both a parameter and a subparameter. |

| | |
|---|---|
| HELP <*prefix*>* | Display a selection list containing matching parameters only. If the string <*prefix*> is left empty, all parameters are displayed. |
| HELP HELP | Display the categories ADD-ON, DRIVER and MAIN on a separate menu. Select a category to display a list of related parameters. You can then select a parameter from this list to view the corresponding help map. |
| HELP <*category*> | List all parameters related to a category from above. |

For further instructions on how to use a help screen invoked this way, see ***Help with Parameters***.

# XX


- 100 Natural Profiler ..... 753
- 101 Profiling Natural Applications ..... 755
- 102 Code Coverage of Natural Applications ..... 767
- 103 Basic Concepts of the Profiler Utility ..... 779
- 104 Using the Profiler Utility in Online Mode ..... 797
- 105 Using the Profiler Utility in Batch Mode ..... 811
- 106 Natural Profiler MashApp ..... 893

# 100 Natural Profiler

This document provides information on profiling Natural applications in order to analyze program execution and code coverage.

| | |
|---|---|
| **Profiling Natural Applications** | General information on the profiling options provided by Natural and NaturalONE. |
| **Code Coverage of Natural Applications** | General information on the options for code coverage provided by Natural and NaturalONE. |
| **Basic Concepts of the Profiler Utility** | Basic concepts of the Profiler utility in online mode and batch mode. |
| **Using the Profiler Utility in Online Mode** | Profiling Natural online programs. |
| **Using the Profiler Utility in Batch Mode** | Profiling and code coverage of Natural batch programs. |
| **Natural Profiler MashApp** | Evaluating Profiler data on an interactive MashZone dashboard. |

**Note:** The features of the NaturalONE Profiler and NaturalONE code coverage are described in the relevant sections of the NaturalONE documentation.

# 101 Profiling Natural Applications


- Introducing Profiling ..... 756
- Platform-Specific Profiling ..... 756
- Profiling Tools ..... 757
- Natural Profiler Evaluations ..... 759

## Introducing Profiling

A profiler is a tool for dynamic program analysis. It measures the frequency and duration of instructions to simplify program optimization.

The Natural Profiler is used to profile Natural applications. It collects profiling data whenever a defined Natural event occurs, for example, when a program starts or before a database is called. The Natural Profiler visualizes the recorded event data as an event trace and the calling structure of the executed Natural objects as a program trace. The performance evaluation provided by the Natural Profiler shows the time consumption and hit count of the executed objects, Natural statements and program lines.

You can view Natural Profiler event data in the Profiler utility output or export the data in text or table format. You can visualize Natural Profiler performance analyses in NaturalONE (Software AG's Eclipse-based development environment) or MashZone (Software AG's tool for creating interactive business dashboards).

A Natural Profiler analysis serves as the basis for performance optimization of a Natural application. The Natural Profiler provides you with a very fast overview about the time-consuming parts of a Natural application. No code modification is required, and moreover, just basic knowledge of the application is sufficient.

## Platform-Specific Profiling

You can profile Natural applications on UNIX, Windows and mainframe platforms. How to profile a Natural application depends on the platform and the application processing mode used:

**Mainframes**

- Mainframe interactive applications are profiled with the NaturalONE Profiler or the Profiler utility in online mode.
- Mainframe interactive applications executed remotely from Natural Studio or RPC are profiled with the Profiler utility in batch mode.
- Mainframe batch applications are profiled with the Profiler utility in batch mode.

**UNIX and Windows**

- UNIX and Windows interactive applications are profiled with the NaturalONE Profiler or the Natural Profiler for UNIX and Windows, respectively.
- UNIX and Windows batch applications are profiled with the Natural Profiler for UNIX and Windows, respectively.

# Profiling Tools

This section summarizes the key features of the Natural profiling tools:

- Features of the NaturalONE Profiler
- Features of the Natural Profiler for UNIX and Windows
- Features of the Profiler Utility
- Features of the Natural Profiler MashApp

## Features of the NaturalONE Profiler

- Profiles interactive Natural applications from UNIX, Windows or mainframe platforms in an Eclipse-based development environment.
- Reads and analyzes Profiler resource files containing event data collected by the mainframe Profiler utility in batch mode or by the Natural Profiler for UNIX and Windows.
- Provides features for big data handling:
  - Event filter,
  - Sampling technique,
  - Data consolidation.
- Performance analyses of programs, statements and program lines:
  - CPU time,
  - Elapsed time,
  - Hit count.
- Displays an event trace.
- Provides direct navigation from a profiled program line to the corresponding source code.
- Saves and reloads the Profiler data as an XML-formatted file.

## Features of the Natural Profiler for UNIX and Windows

- Profiles interactive or Natural batch applications from UNIX or Windows platforms.
- Provides features for big data handling:
  - Event filter,
  - Sampling technique,
  - Data consolidation.
- Saves the Profiler data as a Profiler resource file.

## Features of the Profiler Utility

### Online Mode (Mainframes)

- Profiles interactive Natural applications from mainframe platforms.
- Provides an event filter.
- Displays an event trace.
- Saves the Profiler data in a table format.
- Saves the Profiler data as a Profiler resource file.

**Note:** The amount of data collected by the Profiler utility in online mode is restricted by the relatively small size of the Natural Data Collector buffer which works in a wrap-around mode. Moreover, when running under CICS or Com-plete, the CPU time is not provided. In general, we recommend that you use the NaturalONE Profiler for profiling interactive Natural mainframe applications because the NaturalONE Profiler has no size restrictions and supports CPU performance analyses.

### Batch Mode (Mainframes)

- Profiles Natural batch and Natural RPC applications from mainframe platforms.
- Profiles mainframe interactive applications executed remotely from Natural Studio.
- Provides features for big data handling:
  - Event, program, count and time filters,
  - Sampling technique,
  - Data consolidation.
- Saves Profiler data as a Profiler resource file.
- Reads and analyzes Profiler resource files.
- Prints program and event traces.
- Analyzes program performance.
- Collects and displays Profiler properties and statistics.
- Exports Profiler data for MashZone visualization.

### Batch Mode (UNIX and Windows)

- Reads and analyzes Profiler resource files.
- Provides features for big data handling:
  - Data consolidation.
- Saves consolidated Profiler data as a Profiler resource file.
- Prints program and event traces.
- Analyses program performance.

- Displays Profiler properties and statistics.
- Exports Profiler data for MashZone visualization.

### Features of the Natural Profiler MashApp

- Visualizes Profiler data on a graphical, interactive MashZone dashboard.
- Analyzes application performance with selection criteria such as library, program, program line and user:
  - CPU time,
  - Elapsed time,
  - Adabas command time,
  - Hit count.
- Displays Profiler properties and statistics.

## Natural Profiler Evaluations

This section describes the evaluations provided by the Natural profiling tools:


- Performance Analysis
- Program Trace
- Event Trace
- Profiler Properties and Statistics


### Performance Analysis

You can analyze program performance with the Natural Profiler to identify critical sections of source code.

**Program Summary Report**

The **Natural Profiler Program Summary** report of the Profiler utility shows the CPU time spent during Natural program execution. It also shows which and how many Natural events occurred.

Example:

```
Natural Profiler Program Summary
--------------------------------
Library  Program  Start    Load  Database  I/O  External  Error Statement  User CPU-Time (ms)  CPU %
SYSEDMD  ADA-CL      41       0        40    0        41      0       621     0         3.785   0.14
SYSEDMD  ADA-RC      45       0        44    0        45      0       545     0         4.704   0.17
SYSEDMD  AOS-CL     115      97        15    0         0      0      2507     0        42.890   1.63
SYSEDMD  AOS-OP     169     154        22    0         0      0      6975     0        70.286   2.68
SYSEDMD  MONADA       1    3176        71    0         0      0    272180     0       680.214  25.98
SYSEDMD  SPSTATE      7       0         8    0         0      1        84     0         1.144   0.04
SYSLIBS  A82FCB    2797       0         0    0      2797      0     39314     0       258.930   9.89
...
Total              5294    5293      2122    7      6510     43    857384     0      2617.326 100.00
```

In the extract of a **Program Summary** above, the `MONADA` program used `25.98` percent of the CPU time (680 ms) and executed `272180` statements. It was started once, loaded `3176` other Natural objects and issued `71` database calls. The program `ADA-RC` called `45` external programs. One error occurred in the program `SPSTATE`.

**Hot Spots Page**

The **Hot Spots** page of the NaturalONE Profiler shows the CPU time and the elapsed time used by Natural objects, statements and program lines. It also shows how often an object or statement executed. From a profiled program line you can directly navigate to the corresponding source code line.

Example:

BENCHI [PRFDEMO]

| Hot Spot | Elapsed Time | CPU ... | Hit Count |
|---|---|---|---|
| BENCHI (PRFDEMO) | | | |
| EXAMINE | | | |
| Line 1420 | 0,4% | 32,9% | 1000 |
| Line 1280 | < 0,1% | 2,9% | 1000 |
| CALLNAT | 0,4% | 20,2% | 1000 |
| ASSIGN/COMPUTE/MOVE | 0,4% | 16,4% | 9105 |
| FOR | 0,3% | 9,5% | 6012 |
| LOOP | 0,3% | 7,7% | 6000 |
| RESIZE | < 0,1% | 2,4% | 1001 |
| IGNORE | < 0,1% | 1,0% | 1000 |
| INPUT | 97,8% | 0,6% | 1 |
| WRITE | < 0,1% | 0,3% | 8 |
| INCLUDE | < 0,1% | < 0,1% | 28 |
| IF | < 0,1% | < 0,1% | 27 |
| Initialization | < 0,1% | < 0,1% | 1 |
| PERFORM | < 0,1% | < 0,1% | 12 |
| RETURN | < 0,1% | < 0,1% | 12 |
| RESET | < 0,1% | < 0,1% | 6 |
| END | < 0,1% | < 0,1% | 1 |
| Internal Statement Code FF80 | < 0,1% | < 0,1% | 5 |
| Internal Statement Code FF84 | < 0,1% | < 0,1% | 2 |
| BENCH-C (PRFDEMO) | 0,2% | 5,9% | 1000 |

Hot Spots | Event Trace

In the example above, the highest percent of CPU time was used by the `EXAMINE` statement in Line `1420` in the `BENCHI` program which was executed `1000` times. If you double-click on the entry of `Line 1420`, the NaturalONE editor opens the program source of `BENCHI` and positions on the corresponding line.

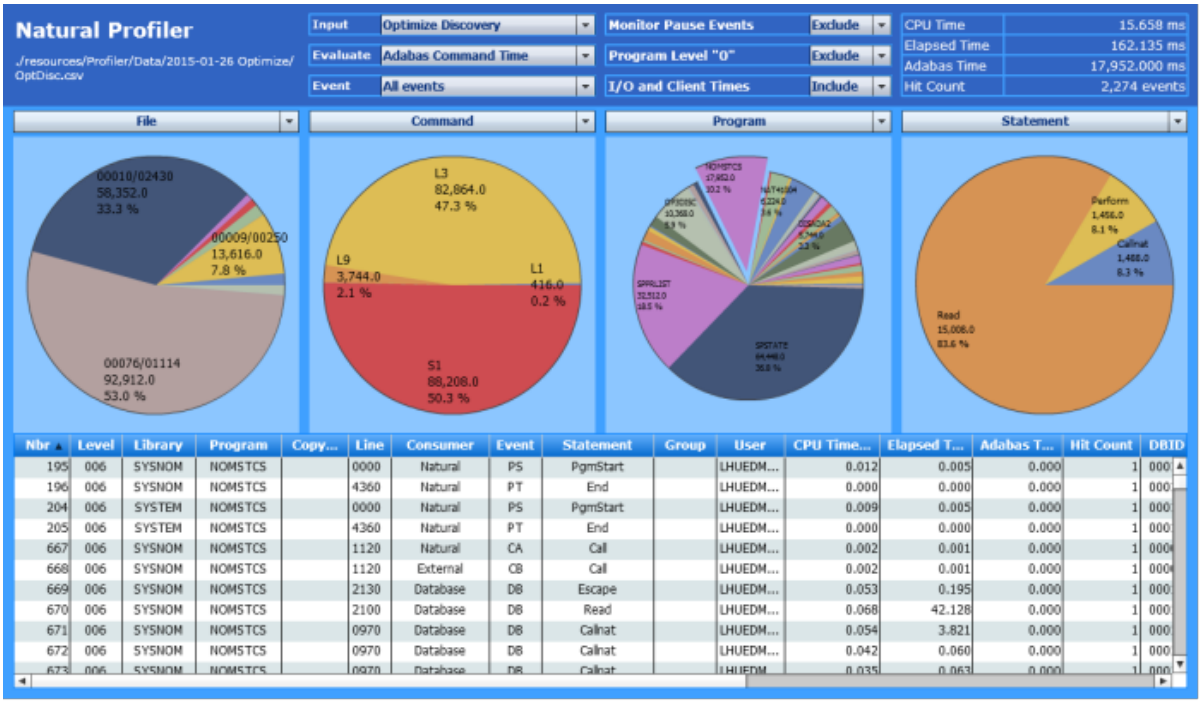
**Evaluation Page**

The **Evaluation** page of the **Natural Profiler MashApp** visualizes the Profiler data on an interactive MashZone dashboard. You can evaluate the distribution of the CPU time, the elapsed time, the Adabas command time or the hit count using the following criteria:

- Consumer (such as Natural or a database )
- Event

- Group (group of users as defined in Natural Security)
- Level (program execution level)
- Library
- Line
- Line100 (lines with similar line number)
- Program
- Statement
- User (for Natural RPC applications running under Natural Security)
- Client User
- Command (Adabas database call)
- File
- Return Code
- Target Program
- Type (for example, object type subprogram for a program start event)

Example:




The pie charts in the example above show the distribution of the Adabas command time (selected in the header) for the following criteria (selected above the pie charts):

1. Adabas files,

2. Adabas commands,
3. Natural programs, and
4. Natural statements.

In the pie chart below **Program**, the `NOMSTCS` program has been selected by clicking on the corresponding segment. Therefore, the pie chart below **Statement** shows the distribution of the Adabas command time over the statements of the `NOMSTCS` program only.

### Program Trace

The **Program Trace** provided by the Natural Profiler utility shows the program flow of the profiled application. It lists all program events and shows which and how many events occurred between the program events.

Example:

```
Natural Profiler Program Trace
------------------------------
Time            Ev Library  CC-Name  Line Lev Program  Events
10:20:58.309812 PL                   0000 000
10:20:58.311790 PR SYSEDMD           1210 001 .OPTTEST  D=3 N=30
10:20:58.322550 PL SYSEDMD           7200 001 .OPTTEST
10:20:58.322573 PS SYSEDMD           0000 002 ..PROFREV  D=5 N=18
10:20:58.323566 PL SYSEDMD           1040 002 ..PROFREV
10:20:58.323572 PS SYSLIBS           0000 003 ...NAT41004 D=3 C=6 N=344
10:20:58.335128 PT SYSLIBS           5235 003 ...NAT41004
10:20:58.335128 PR SYSEDMD           1040 002 ..PROFREV  N=162
10:20:58.335260 PT SYSEDMD           2180 002 ..PROFREV
10:20:58.335260 PR SYSEDMD           7200 001 .OPTTEST  D=3 I=4 N=128
...
10:21:41.731229 PT SYSEDMD           7370 001 .OPTTEST
10:21:41.731229 PR                   0000 000          D=14 I=1
10:21:42.248348 ST                   0000 000

Totals
------
Ev Event                    Count
S  Session ................ 1
P  Program ................ 5297
D  Database Call .......... 2140
I  Terminal I/O ........... 12
C  External Program Call .. 6510
E  Runtime Error .......... 43
N  Natural Statement ...... 857384
R  RPC Request............. 0
U  User-Defined Event ..... 0
M  Monitor Pause .......... 2
```

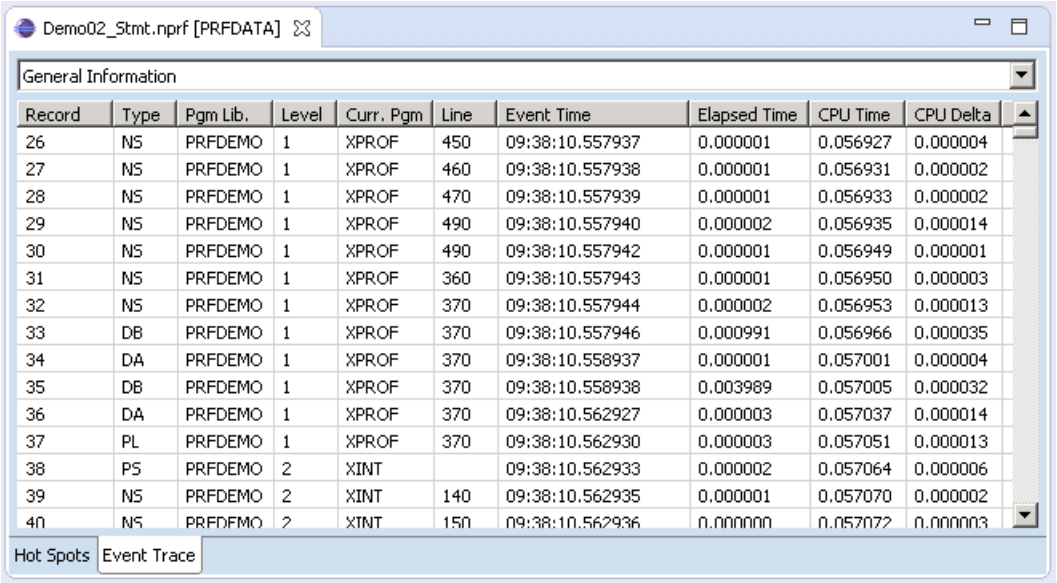
In the example above, the `OPTTEST` program calls `PROFREV` which then calls `NAT41004`.

In Line `1210`, `OPTTEST` performed a Program Resume event (`PR`). It then executed 30 Natural statements (`N=30`) and 3 database calls (`D=3`) before it loaded `PROFREV` indicated by a Program Load (`PL`) event. The `Totals` show how often the events occurred during the entire Natural session.

### Event Trace

The **Event Trace** provided by the NaturalONE Profiler and by the Natural Profiler utility lists the recorded event data in chronological order.

Example:



Demo02_Stmt.nprf [PRFDATA]

General Information

| Record | Type | Pgm Lib. | Level | Curr. Pgm | Line | Event Time | Elapsed Time | CPU Time | CPU Delta |
|---|---|---|---|---|---|---|---|---|---|
| 26 | NS | PRFDEMO | 1 | XPROF | 450 | 09:38:10.557937 | 0.000001 | 0.056927 | 0.000004 |
| 27 | NS | PRFDEMO | 1 | XPROF | 460 | 09:38:10.557938 | 0.000001 | 0.056931 | 0.000002 |
| 28 | NS | PRFDEMO | 1 | XPROF | 470 | 09:38:10.557939 | 0.000001 | 0.056933 | 0.000002 |
| 29 | NS | PRFDEMO | 1 | XPROF | 490 | 09:38:10.557940 | 0.000002 | 0.056935 | 0.000014 |
| 30 | NS | PRFDEMO | 1 | XPROF | 490 | 09:38:10.557942 | 0.000001 | 0.056949 | 0.000001 |
| 31 | NS | PRFDEMO | 1 | XPROF | 360 | 09:38:10.557943 | 0.000001 | 0.056950 | 0.000003 |
| 32 | NS | PRFDEMO | 1 | XPROF | 370 | 09:38:10.557944 | 0.000002 | 0.056953 | 0.000013 |
| 33 | DB | PRFDEMO | 1 | XPROF | 370 | 09:38:10.557946 | 0.000991 | 0.056966 | 0.000035 |
| 34 | DA | PRFDEMO | 1 | XPROF | 370 | 09:38:10.558937 | 0.000001 | 0.057001 | 0.000004 |
| 35 | DB | PRFDEMO | 1 | XPROF | 370 | 09:38:10.558938 | 0.003989 | 0.057005 | 0.000032 |
| 36 | DA | PRFDEMO | 1 | XPROF | 370 | 09:38:10.562927 | 0.000003 | 0.057037 | 0.000014 |
| 37 | PL | PRFDEMO | 1 | XPROF | 370 | 09:38:10.562930 | 0.000003 | 0.057051 | 0.000013 |
| 38 | PS | PRFDEMO | 2 | XINT | | 09:38:10.562933 | 0.000002 | 0.057064 | 0.000006 |
| 39 | NS | PRFDEMO | 2 | XINT | 140 | 09:38:10.562935 | 0.000001 | 0.057070 | 0.000002 |
| 40 | NS | PRFDEMO | 2 | XINT | 150 | 09:38:10.562936 | 0.000000 | 0.057072 | 0.000003 |

Hot Spots | Event Trace

The example above shows general event data such as the library or program name. In addition to the recorded event and CPU timestamps, the NaturalONE Profiler displays the corresponding delta values (elapsed time and CPU delta, respectively). You can also view event-specific data such as the number of the file used for database calls.

### Profiler Properties and Statistics

The **Profiler properties and statistics** provided by the Natural Profiler utility and the **Natural Profiler MashApp** lists Profiler properties such as the Profiler revision, and statistics of the monitored application that show, for example, the total CPU time and the elapsed time.

Example:

**Natural Profiler**

./resources/Profiler/Data/2015-04-27 Optimize/OptMoni.csv

Input: 2015-04-27 Optimize Monitor

Category: All categories

**Application CPU time**
The total CPU time consumed by the application.

| Seq | | Category | Property | Value | Unit |
|---|---|---|---|---|---|
| 25 | | Monitor Session | Monitor start time | 2015-04-29 10:07:57.4 | |
| 26 | | Monitor Session | Monitor end time | 2015-04-29 10:08:22.0 | |
| 27 | | Monitor Session | Monitor elapsed time | 24.593283 | sec |
| 28 | | Trace Session | First library | SYSEDM | |
| 29 | | Trace Session | First program | MENU | |
| 30 | | Trace Session | Highest level | 10 | |
| 31 | | Trace Session | Trace start time | 10:07:58.314436 | |
| 32 | | Trace Session | Trace end time | 10:08:21.687841 | |
| 33 | | Trace Session | Trace elapsed time | 23.373405 | sec |
| 34 | | Trace Session | Application CPU time | 1626.441 | ms |
| 35 | | Trace Session | Monitor CPU time | 295.919 | ms |
| 36 | | Trace Session | Total CPU time | 1922.360 | ms |
| 37 | | Trace Session | Sampling interval | 0 | microsec |
| 38 | | Trace Session | Data pool empty | 14 | |
| 39 | | Trace Session | Data pool empty after full | 0 | |
| 40 | | Trace Session | Data pool overflow | 0 | |
| 41 | | Trace Session | No session active | 0 | |
| 42 | | Data Processing | Number of events | 69988 | |
| 43 | | Data Processing | Highest event number | 69985 | |
| 44 | | Data Processing | Number of data blocks | 19 | |
| 45 | | Data Processing | Utility buffer size | 5000 | bytes |
| 46 | | Data Processing | Data block size | 4974 | bytes |
| 47 | | Data Processing | RDC data length | 13354208 | bytes |
| 48 | | Data Processing | Uncompressed data length | 288919 | bytes |
| 49 | | Data Processing | Compressed data length | 80318 | bytes |
| 50 | | Data Processing | Identical bytes trimmed left | 201163 | |
| 51 | | Data Processing | Blanks trimmed right | 7438 | |

The example above shows statistics of the monitor session, the trace session and the data processing. The monitor session was running for `24.59` seconds, the application consumed `1626.441` microseconds of CPU time, and the Profiler recorded `69988` events.

# 102 Code Coverage of Natural Applications


- Introducing Code Coverage ..... 768
- Platform-Specific Code Coverage ..... 769
- Code Coverage Tools ..... 769
- Natural Code Coverage Evaluations ..... 772

This document provides general information on code coverage of Natural applications.

## Introducing Code Coverage

In general, code coverage measures the degree to which the source code of a program is executed. It is often used for systematic software testing. The higher the code coverage percentage is, the lower is the chance that the code contains undetected software bugs in code that is not executed.

The Natural code coverage is used to monitor the executed statements of a Natural application. It collects the coverage data while the application is executed and provides tools to analyze the collected data afterwards.

The **Code Coverage** view of NaturalONE (Software AG's Eclipse-based development environment) and the **Program Coverage** table of the Profiler utility show - expressed as a percentage - how many of the statements of the Natural objects have been executed. The Natural Coverage Plugin for Jenkins visualizes the outcome of a Natural coverage cycle directly in the Jenkins job result pages.

In the NaturalONE editor and in the **Statement Coverage** table of the Profiler utility you can see, which individual statement lines of a Natural object have been executed. Here you can also see the statement lines which have been missed or have only been partly covered.

If a statement source uses multiple lines, only the line in which the statement begins is mentioned in the coverage reports.

You can export the coverage data with the Profiler utility output in text or table (CSV) format. The CSV table can be analyzed with a spreadsheet software such as Microsoft Excel.

### GP and Source Coverage

If a Natural source contains `INCLUDE` statements, the corresponding copycode is included in the generated object (the **GP**). For the coverage, we can monitor two statement counts:

1. The number of the statements in the GP which includes all copycodes recursively (a copycode can include further copycodes).
2. The number of the statements in the source which does not include the copycodes.

The GP coverage reflects the percentage of the covered statements in the GP including copycodes; whereas the source coverage reflects the percentage of the covered statements in the source not including copycodes.

## Platform-Specific Code Coverage

You can perform the Natural code coverage on UNIX, Windows and mainframe platforms. How to proceed depends on the platform and the application processing mode used:

### Mainframes

- Code coverage of mainframe interactive applications remotely executed from Natural Studio or RPC is performed using the Profiler utility in batch mode.
- Code coverage of mainframe batch applications is performed using the Profiler utility in batch mode.

**Note:** Code coverage is not available for mainframe interactive applications running locally on a mainframe or remote from NaturalONE.

### UNIX and Windows

- Code coverage of UNIX and Windows interactive applications is performed with the NaturalONE code coverage or the Natural code coverage for UNIX and Windows, respectively.
- Code coverage of UNIX and Windows batch applications is performed with Natural code coverage for UNIX and Windows, respectively.

## Code Coverage Tools

This section summarizes the key features of the Natural profiling tools:

- Features of the NaturalONE Code Coverage
- Features of the Natural Code Coverage for UNIX and Windows
- Features of the Profiler Utility

- Features of the Natural Code Coverage Spreadsheet

### Features of the NaturalONE Code Coverage

- Reads and analyzes Natural code coverage resource files containing coverage data collected by the mainframe Profiler utility in batch mode or by the Natural code coverage for UNIX or Windows.
- Interactive Natural applications from UNIX or Windows can be covered by activating the Natural code coverage for UNIX or Windows and reading the corresponding Natural code coverage resource file.
- The Natural code coverage view shows which percentage of the statements of the Natural objects have been executed.
- From the Natural code coverage view the involved Natural objects can be edited. The NaturalONE editor displays all covered lines with a green background.

**Note:** Interactive code coverage of Natural applications from mainframe platforms is currently not supported.

### Features of the Natural Code Coverage for UNIX and Windows

- Code coverage of interactive or Natural batch applications from UNIX or Windows platforms.
- Provides features for big data handling:
  - Automatic event filter,
  - automatic data consolidation.
- Saves the code coverage data as a Natural code coverage resource file.

### Features of the Profiler Utility

#### Batch Mode (Mainframes)

- Code coverage of Natural batch applications from mainframe platforms.
- Code coverage of mainframe interactive applications remotely executed from Natural Studio or against a Natural RPC server.
- Provides features for big data handling:
  - Program, count and time filters,
  - automatic event filter,
  - automatic data consolidation.
- Saves code coverage data as a Natural code coverage resource file.
- Reads and analyzes Natural code coverage resource files.

- Lists the **Program Coverage** table for all accessed Natural objects with
  - percentage of the covered statements,
  - number of covered statements,
  - number of missed (not covered) statements and
  - total number of statements of the object.
- The **Statement Coverage** lists the source of each accessed Natural objects and shows for each line the percentage of the covered statements.
- Exports Natural code coverage data in CSV (comma-separated values) format which can be further analyzed with a spreadsheet software (e.g. Microsoft Excel).
- Collects and displays Profiler and code coverage properties and statistics.

**Note:** On the mainframe, there is no one-to-one relationship between a Natural source code statement and the corresponding object code in the cataloged object. The Natural code coverage on the mainframe monitors the object code rather than the Natural source code. Therefore, multiple Natural statements can be merged into one coverage entry and conversely, one Natural statement can cover multiple coverage entries.

**Batch Mode (UNIX and Windows)**

- Reads and analyzes Natural code coverage resource files.
- Lists the **Program Coverage** table for all accessed Natural objects with
  - percentage of the covered statements,
  - number of covered statements,
  - number of missed (not covered) statements and
  - total number of statements of the object.
- The **Statement Coverage** lists the source of each accessed Natural objects and shows for each line the percentage of the covered statements.
- Exports Natural code coverage data in CSV (comma-separated values) format which can be further analyzed with a spreadsheet software (e.g. Microsoft Excel).

**Note:** On Windows and UNIX, missed statements are not collected. Therefore the Statement Coverage can only mark lines containing covered statements and the coverage of these lines is always 100%.

### Features of the Natural Code Coverage Spreadsheet

- Template for coloring the Natural code coverage data exported in CSV (comma-separated values) format by the Natural Profiler utility.
- Program and copycode coverage with source and GP counters for
  - percentage of the covered statements,
  - number of covered statements,
  - number of missed (not covered) statements and
  - total number of statements of the object.
- Statement Coverage of the object source whereby the lines are colored in
  - green – if all statements of the line are covered,
  - yellow – if the statements of the line are partly covered,
  - red – if all statements of the line are missed,
  - gray – if the line is empty or contains only comments.
- Profiler and code coverage properties and statistics (for mainframe data).

**Note:** A Microsoft Excel spreadsheet template for Natural code coverage is available as a resource in the Natural Profiler library `SYSPRFLR` on UNIX and Windows.

## Natural Code Coverage Evaluations

This section describes the evaluations provided by the Natural code coverage tools:

- Program Coverage
- Line and Statement Coverage
- Profiler Properties and Statistics

### Program Coverage

The program coverage provides you with an overview of the programs executed and the amount of the code that has been covered by the application.

### Program Coverage Report

The **Program Coverage** report of the Profiler utility shows the coverage (in percentage of the total number of statements) of each Natural object executed. It shows for each object how many statements have been covered or missed and the total number of statements. In addition, it summarizes the values for all objects in a library and the totals over all libraries.

If the output is written in text format, only the GP coverage is provided. If the data is exported in CSV (comma-separated values) format, the source coverage is given as well. Additionally, the counters for all included copycodes are printed.

The following is an example for text format:

```
Program Coverage
----------------
Library  Object   Ty Coverage%  Covered    Missed    Total
COVDEMO  TESTCOVN N      84.0%       37         7       44
COVDEMO  TESTCOVP P      69.2%        9         4       13
COVDEMO  -------- --     80.7%       46        11       57
Totals   -------- --     80.7%       46        11       57
```

In the **Program Coverage** example above, 69.2% of 13 statements in the `TESTCOVP` program were covered, corresponding to 9 covered and 4 missed statements. 80.7% of the statements of the accessed objects in the library `COVDEMO` were covered, which is also the total value for the whole application run.

### Code Coverage View

The **Code Coverage** view of NaturalONE shows the coverage (in percentage of the total number of statements) of each Natural object executed. It shows for each object how many statements have been covered or missed and the total number of statements. If copycodes are included, the object node can be opened to view the coverage of the copycode. In general, the counters reflect the GP coverage (copycodes included). The source coverage (copycodes not included) is displayed in the line where the object name is enclosed in the `<< >>` brackets.

From any line you can directly navigate to the corresponding source code to view the statement coverage.

Example:

Console Problems Tasks Details Error Log Code Coverage

| Object | Coverage | Covered | Missed | Total |
|---|---|---|---|---|
| TESTCOVP (COVDEMO) | 69,2% | 9 | 4 | 13 |
| TESTCOVC (COVDEMO) | 20,0% | 1 | 4 | 5 |
| <<TESTCOVP (COVDEMO)>> | 100,0% | 8 | 0 | 8 |
| TESTCOVN (COVDEMO) | 84,1% | 37 | 7 | 44 |

In the example above, the `TESTCOVP` program has a GP coverage of 69.2 percent whereby in the program itself all 8 statements are covered (100% source coverage) and in the included copycode `TESTCOVC` only 1 of 5 statements was covered.

## Line and Statement Coverage

The statement coverage shows which lines of the program have been executed. For mainframe data, the Profiler utility also indicates which lines containing statements have not been executed or are only executed partly.

### Statement Coverage Report

The **Statement Coverage** report of the Profiler utility shows for each source line the coverage of the statements in the line. If the data is exported in CSV (comma-separated values) format, the number of covered or missed statements and the total number of statements in the line are printed as well. The Microsoft Excel spreadsheet template delivered with Natural on UNIX and Windows, can be used to color the lines according to the coverage.

If a source contains an `INCLUDE` statement, the corresponding copycode source is included in the report right after the `INCLUDE` statement.

The following is an example for an export in CSV format colored using a Microsoft Excel spreadsheet:

| Line | Source | Coverage% | Covered | Missed | Total |
|---|---|---|---|---|---|
| 10 | * Test function Coverage | | 0 | 0 | 0 |
| 20 | * Subprogram TESTCOVN | | 0 | 0 | 0 |
| 30 | DEFINE DATA | | 0 | 0 | 0 |
| 40 | PARAMETER | | 0 | 0 | 0 |
| 50 | 1 FUNC (I2) /* function | | 0 | 0 | 0 |
| 60 | 1 RET-CODE (I4) /* Return code | | 0 | 0 | 0 |
| 70 | END-DEFINE | | 0 | 0 | 0 |
| 80 | * | | 0 | 0 | 0 |
| 90 | /* Return 0 by default | | 0 | 0 | 0 |
| 100 | RESET RET-CODE | 100 | 1 | 0 | 1 |
| 110 | * | | 0 | 0 | 0 |
| 120 | DECIDE ON FIRST VALUE OF FUNC | 100 | 1 | 0 | 1 |
| 130 | VALUE 0 | 50 | 1 | 1 | 2 |
| 140 | PRINT 'Test function 0' | 0 | 0 | 1 | 1 |
| 150 | VALUE 1 | 66 | 2 | 1 | 3 |
| 160 | PRINT 'Test function 1' | 100 | 1 | 0 | 1 |
| 170 | VALUE 2 | 100 | 3 | 0 | 3 |
| 180 | PRINT 'Test function 2' | 100 | 1 | 0 | 1 |
| 190 | VALUE 3 | 100 | 3 | 0 | 3 |
| 200 | PRINT 'Test function 3' | 100 | 1 | 0 | 1 |
| 210 | VALUE 4 | 100 | 3 | 0 | 3 |
| 220 | PRINT 'Test function 4' | 100 | 1 | 0 | 1 |
| 230 | VALUE 5 | 100 | 3 | 0 | 3 |
| 240 | PRINT 'Test function 5' | 100 | 1 | 0 | 1 |
| 250 | VALUE 6 | 100 | 3 | 0 | 3 |
| 260 | PRINT 'Test function 6' | 100 | 1 | 0 | 1 |
| 270 | VALUE 7 | 100 | 3 | 0 | 3 |
| 280 | PRINT 'Test function 7' | 100 | 1 | 0 | 1 |
| 290 | VALUE 8 | 100 | 3 | 0 | 3 |
| 300 | PRINT 'Test function 8' | 100 | 1 | 0 | 1 |
| 310 | VALUE 9 | 33 | 1 | 2 | 3 |
| 320 | PRINT 'New test function 9' | 0 | 0 | 1 | 1 |
| 330 | NONE VALUE | 100 | 1 | 0 | 1 |
| 340 | RET-CODE := 1 /* Unsupported function | 0 | 0 | 1 | 1 |
| 350 | END-DECIDE | | 0 | 0 | 0 |
| 360 | * | | 0 | 0 | 0 |
| 370 | END | 100 | 1 | 0 | 1 |

The three red lines of the subprogram `TESTCOVN` have not been executed. Thus the test run does not cover the new test function `9`. It also neither covers the (old) function `0` nor the case when the subprogram is called with an unsupported function.

The data originates from the mainframe. Therefore, the counts refer object code statements rather than Natural statements. A Natural `VALUE` statement can correspond up to 3 object code statements. The yellow lines refer to `VALUE` statements where some of the object code has been covered and some not.

### NaturalONE Source Editor

If the source editor is opened from the **Code Coverage** view in NaturalONE, the source is colored according to code coverage. Every line in which one or more statements are covered, is colored with a green background.

Example:

```
TESTCOVP.NSP   TESTCOVN.NSN   TESTCOVC.NSC
 1 * >Natural Source Header 000000
 6 * Test function Coverage
 7 * Subprogram TESTCOVN
 8 DEFINE DATA
 9 PARAMETER
10 1 FUNC      (I2)  /* function
11 1 RET-CODE (I4)  /* Return code
12 END-DEFINE
13 *
14 /* Return 0 by default
15 RESET RET-CODE
16 *
17 DECIDE ON FIRST VALUE OF FUNC
18   VALUE 0
19     PRINT 'Test function 0'
20   VALUE 1
21     PRINT 'Test function 1'
22   VALUE 2
23     PRINT 'Test function 2'
24   VALUE 3
25     PRINT 'Test function 3'
26   VALUE 4
27     PRINT 'Test function 4'
28   VALUE 5
29     PRINT 'Test function 5'
30   VALUE 6
31     PRINT 'Test function 6'
32   VALUE 7
33     PRINT 'Test function 7'
34   VALUE 8
35     PRINT 'Test function 8'
36   VALUE 9
37     PRINT 'New test function 9'
38   NONE VALUE
39     RET-CODE := 1  /* Unsupported function
40 END-DECIDE
41 *
42 END
43
```

The source editor shows all lines in which at least one statement has been executed with a green background. Therefore, all lines except line 19, 37 and 39 of the `DECIDE` statement have been executed.

### Profiler Properties and Statistics

The **Profiler properties and statistics** provided by the Natural Profiler utility lists Profiler properties such as the Profiler revision, and statistics of the monitored application that show, for example, the total CPU time and the elapsed time. For a code coverage run, it shows also the coverage statistics.

**Example**

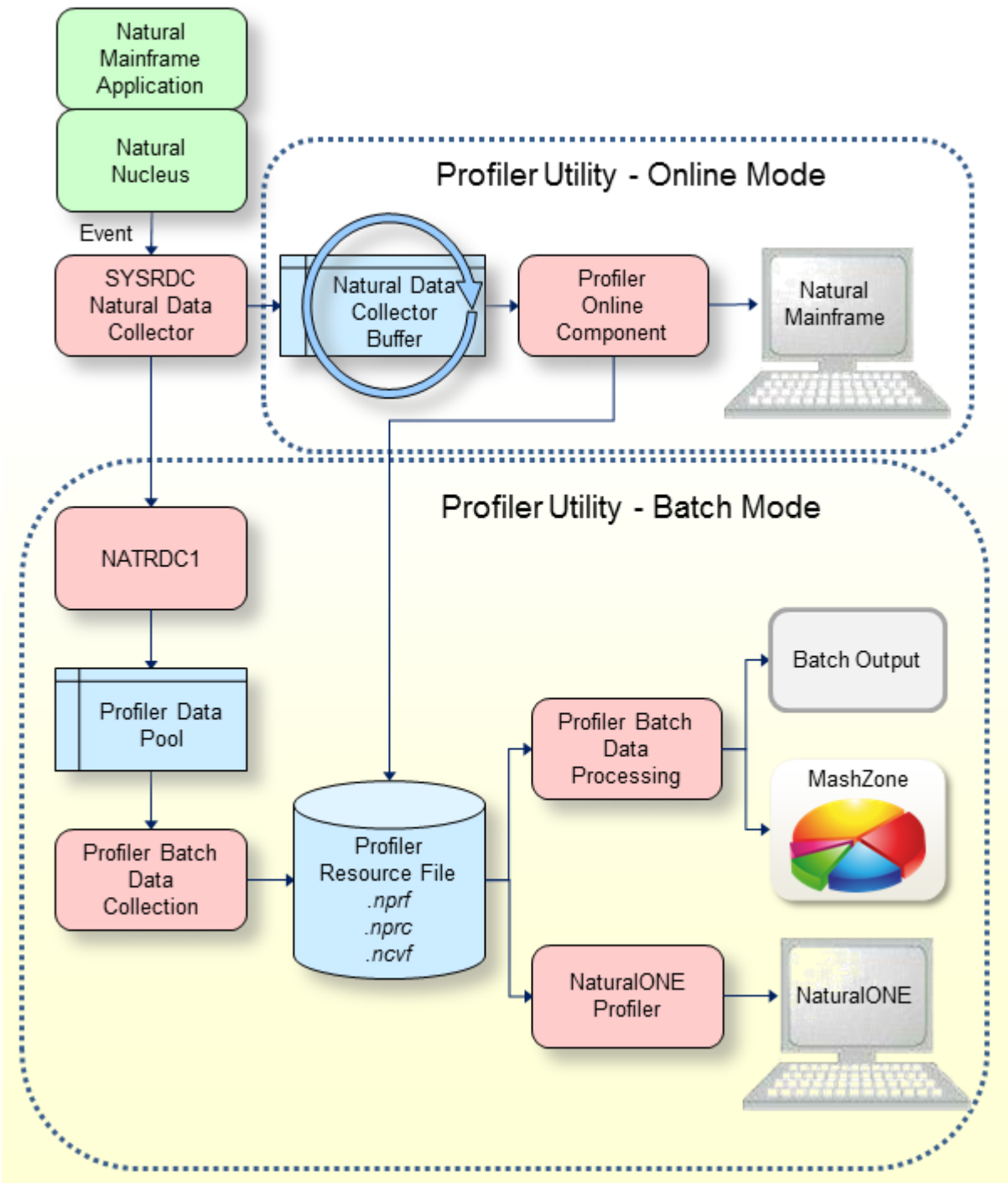
```
************************************************************************
* 13:30:48          ***** NATURAL PROFILER UTILITY *****         2017-09-04
* User SAG                     - Statistics -                       COVREAD
*
* General Info
* Machine class ...................... MAINFRAME
* Environment ........................ Batch
...
* Coverage
* Coverage ........................... ON
* Missed statements recorded ......... ON
* Coverage records ................... 60
* Program information records ........ 3
* Coverage records/block ............. 60
* Bytes/coverage record .............. 10.3
* Programs covered ................... 2
* Statement coverage (percent) ....... 80.7
* Statements covered ................. 46
* Statements missed .................. 11
* Statements total ................... 57
```

# 103 Basic Concepts of the Profiler Utility


- Profiler Utility Overview ..... 780
- Data Collection in Batch ..... 784
- Data Consolidation, Code Coverage and Data Processing ..... 785
- Sampling ..... 789
- Profiler Performance in Batch ..... 791
- Profiling Long-Running Applications ..... 792
- Related Topics ..... 796

The Profiler utility is available for interactive (online) and batch Natural applications. Natural interactive applications can also be profiled from NaturalONE which is described in detail in the *NaturalONE* documentation.

## Profiler Utility Overview




The graphic illustrates the process flow when the Profiler utility traces data in online or batch mode.

The Profiler utility is based on SYSRDC technology, as shown in the graphic. When an event such as a program start occurs in a Natural mainframe application, the Natural nucleus calls the Natural Data Collector of the SYSRDC utility which collects the Natural event data in the Natural Data

Collector buffer and passes the event data to user exits of the Natural Data Collector. The way in which the event data is further processed depends on the mode in which you execute the Profiler utility.

The graphic is further explained in the following section:

**Profiler Utility - Online Mode**

The Profiler utility in online mode is menu-based. It is used for a quick view of the last actions of an interactive Natural application.

- The **Natural Data Collector Buffer** (maximum size of 128 KB) works in a wrap-around mode. It provides the most recent events whereas the oldest data is overwritten when the buffer is full. The data collection is stopped when the data is read and the buffer is cleared when the collection is restarted.
- The **Profiler Online Component** reads the event data from the Natural Data Collector buffer. It provides functions to control profiler tracing, to select required event types, to maintain and display trace records, to download the event data to the PC and to save the event data as Profiler resource file. The Profiler resource file can be processed by the Profiler batch data processing functions and by NaturalONE.

**Profiler Utility - Batch Mode**

The Profiler utility in batch mode is controlled by JCL input. It is designed for analyzing Natural batch applications.

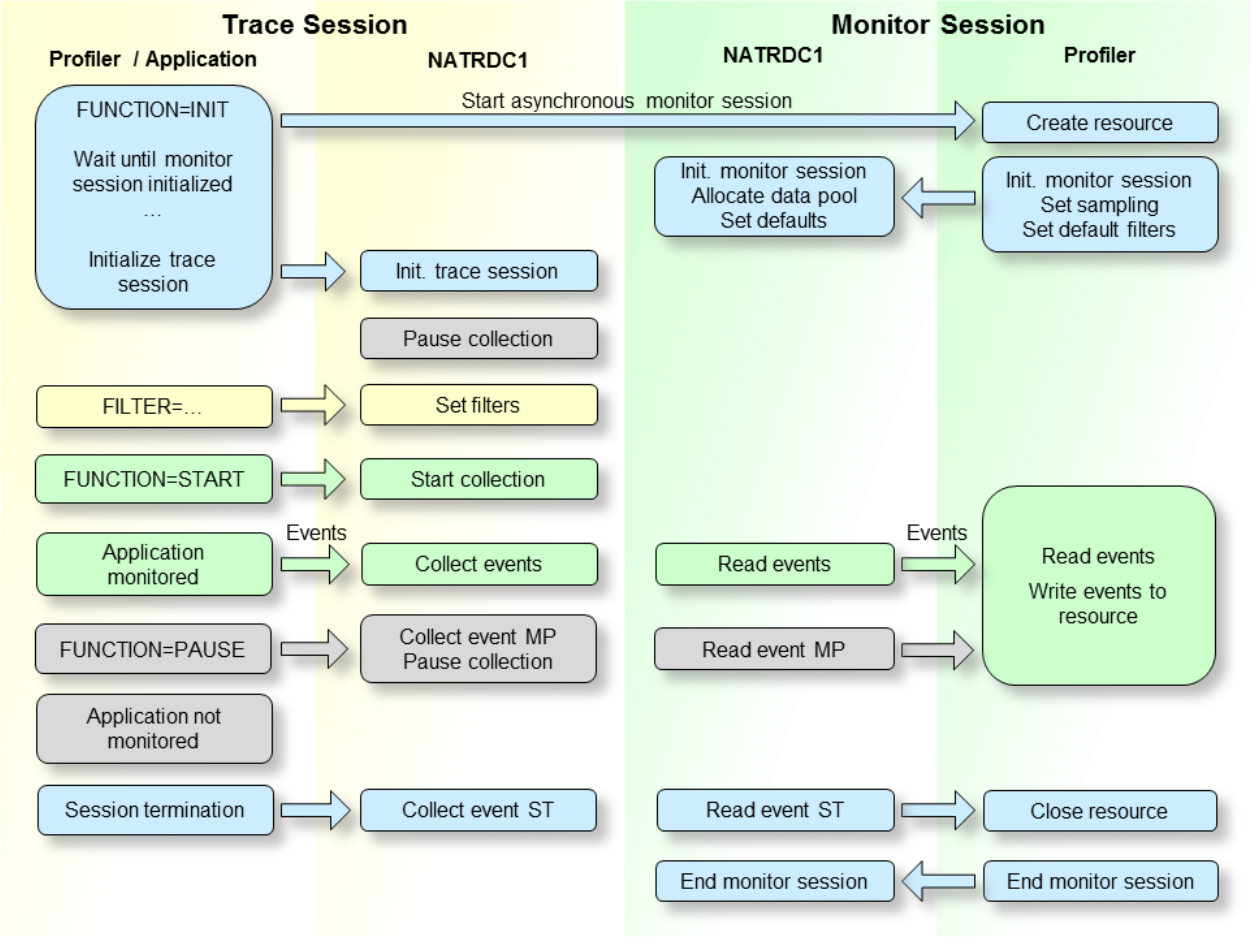
- The user exit **NATRDC1** of the Natural Data Collector collects the Natural event data in the Profiler data pool. It extents the Natural Data Collector event record with additional event information and performs special functions such as filtering or sampling.
- The **Profiler Data Pool** (maximum size 2 GB) collects the Natural event data for the Profiler utility in batch mode and for the NaturalONE Profiler. A special logic guarantees that no data is lost, even if the pool is full.
- The **Profiler Batch Data Collection** reads the event data asynchronously from the Profiler data pool while the application and the tracing continue. It provides functions to control profiler tracing, to select required event types, to filter, sample or consolidate the data, to perform Natural code coverage, and to write the resulting events to a Profiler resource file. General statistics and an event trace with the most important data can be written to standard output.
- The **Profiler Resource File** (extension `.nprf`, `.nprc` or `.ncvf`) is a Natural resource allocated on the FNAT or FUSER system file according the resource library selection. It contains the event data in a compressed format with up to 80 percent data storage reduction. The data is combined in data blocks for an optimized transfer to NaturalONE.
- During Profiler **Batch Data Processing**, the Profiler reads and processes the event data from the Profiler resource files. It provides functions for data consolidation (aggregation), event tracing and program tracing. It offers a program summary and displays the Profiler properties and statistics. For the Natural code coverage data, program and statement coverage reports

are provided. The resulting data can be exported to a file in text or CSV (comma-separated values) format, or in the format expected by the **Natural Profiler MashApp**.

- The Natural Profiler MashApp demonstrates how Profiler event data can be visualized with MashZone.
- In the Natural Server view of NaturalONE, the Profiler resource files are listed as NPRF or NPRC resources. The context menu function **Open with Natural Profiler** reads the resource data to the NaturalONE Profiler. The **NaturalONE Profiler** provides the general analysis of the event data. It shows how the CPU time or elapsed time is distributed over the programs, statements and even program lines of the application and how often a statement was executed. Additionally, the full event trace is provided.
- Coverage resource files (NCVF) can be added from the NaturalONE Server view to a project in the NaturalONE workspace. The context menu function **Open with Natural Code Coverage** reads the resource data and displays it in the **Code Coverage** view.

  From the NaturalONE **Code Coverage** view, the Natural source editor can be opened. It shows all source line with one or more covered statements with a green background.

## Data Collection in Batch




This graphic illustrates which actions the Profiler utility performs when profiling a Natural application in batch mode.

- The Natural trace session is the session in which the Natural batch application is executed and the Profiler trace data is generated and collected.
- When the Profiler utility `INIT` function is performed, a new Natural session is started as an asynchronous subtask. This session is called monitor session because it monitors the events. In both sessions, a Natural nucleus instance with a linked NATRDC1 exit is running. The `INIT` function triggers the execution of the Profiler utility in the monitor session and forwards the `INIT` specific keywords to it.

  In the monitor session, the Profiler resource file is created in which the events will be saved later. Then, the NATRDC1 exit is called to initialize the monitor session. NATRDC1 allocates the Profiler data pool which is used to transfer the event data from the trace session to the monitor session. It also initializes the sampling and sets default values for filters.

- When the Profiler in the trace session notices that the monitor session has been successfully initialized, it calls the NATRDC1 exit to initialize the trace session. By default, the data collection is paused after the initialization.
- After the initialization, the Profiler filters can be set. This should be performed before the data collection is started so that the filters are immediately operative at start.
- Data collection begins with the `START` function. The Profiler in the trace session sends a start request to NATRDC1 which collects the events of the subsequent Natural applications in the Profiler data pool. Simultaneously, the Profiler in the monitor session calls NATRDC1 to read the event data from the Profiler data pool. The space of the event data is immediately released so that the trace session can reuse it. The Profiler in the monitor session compresses the event data and writes it into the Profiler resource file.
- With the `PAUSE` function, data collection can be paused. The Profiler in the trace session sends a pause request to NATRDC1 which writes an `MP` (Monitor Pause) event into the Profiler data pool and suspends data collection, that is, it refuses all events until a start request is received or the trace session ends.
- At the end of the application, an `ST` (session termination) event is written to the Profiler data pool. However, the trace session does not terminate right away (which would abort the monitor session subtask). Instead, it waits for the monitor session to read the remaining data from the Profiler data pool. When the Profiler in the monitor session finds the `ST` event in the Profiler data pool, it closes the Profiler resource file, writes the statistics and sends a termination request to NATRDC1 which ends the monitor session. Finally, the trace session terminates as well.

## Data Consolidation, Code Coverage and Data Processing

The Profiler utility uses technology introduced with the NaturalONE Profiler such as the NATRDC1 user exit and the Profiler data pool. Therefore, the processing of the event data is restricted to NaturalONE users who can use the NaturalONE Profiler and the Profiler utility to evaluate the event data. The data consolidation and processing functions of the Profiler utility (`CONSOLIDATE`, `READ`, `MASHZONE`, `LIST` and `DELETE`) have to be activated before they can be used. The activation is described in ***Prerequisites***.

This section covers the following topics:

- Data Consolidation
- Natural Code Coverage

- Data Processing

## Data Consolidation

When a Natural application is profiled, the Natural Profiler collects one record for each event. Depending on the application, this can produce huge amounts of data, especially when Natural statements are monitored. The more data the Profiler generates, the more time is required to transport the data from the server to the NaturalONE client.

The Profiler utility offers a server-side data consolidation which significantly reduces the amount of data while increasing the transport flow rate. The Profiler data consolidation combines similar records into one consolidated record containing aggregated time values and a hit counter. The consolidated data is written to a resource file which has the same name as the corresponding unconsolidated resource file but an extension `.nprc` (Natural Profiler resource consolidated).

During profiling, the data can be consolidated immediately by switching on the `CONSOLIDATE` keyword of the Profiler utility `INIT` function. Unconsolidated data of an NPRF file can be consolidated later with the Profiler utility `CONSOLIDATE` function.

**Example**

A Natural statement executes 1000 times in a `FOR` loop. The unconsolidated data contains 1000 records for each execution of the statement. Each record contains the event time and the CPU timestamp, besides other information. The Profiler consolidation combines these 1000 records into one consolidated record. All common information (like the library or program name) is kept, the elapsed time and the CPU time of each execution of the statement is determined, summarized and saved in the consolidation record. Additionally, a hit count of `1000` is recorded.

**Notes:**

1. An NPRC resource file that has been consolidated on the server side contains the same hot spot values as the corresponding unconsolidated NPRF resource but opens much faster with NaturalONE.
2. The consolidated data does not contain the event history (timestamps). Therefore, it is not possible to view the event trace when you open an NPRC resource in NaturalONE.
3. Data consolidation is a prerequisite if you want to analyze the event data in MashZone by using the **Natural Profiler MashApp**.

### Natural Code Coverage

Natural code coverage is used to monitor executed and not-executed statements of a Natural application. It is started by switching on the `COVERAGE` keyword of the Profiler utility `INIT` function.

For code coverage, the Profiler automatically uses an event filter so that only the program start (`PS`) and Natural statement (`NS`) events are collected. Moreover, in this case the statement events contain a GP offset which is needed to uniquely identify the statement.

For every accessed program, the corresponding cataloged program (GP) is read and the statement table is build up containing the line number, the GP offset, the object code, and the copycode information of the statement. Additionally, an inverted list of GP offsets is created for a quick search of the offset.

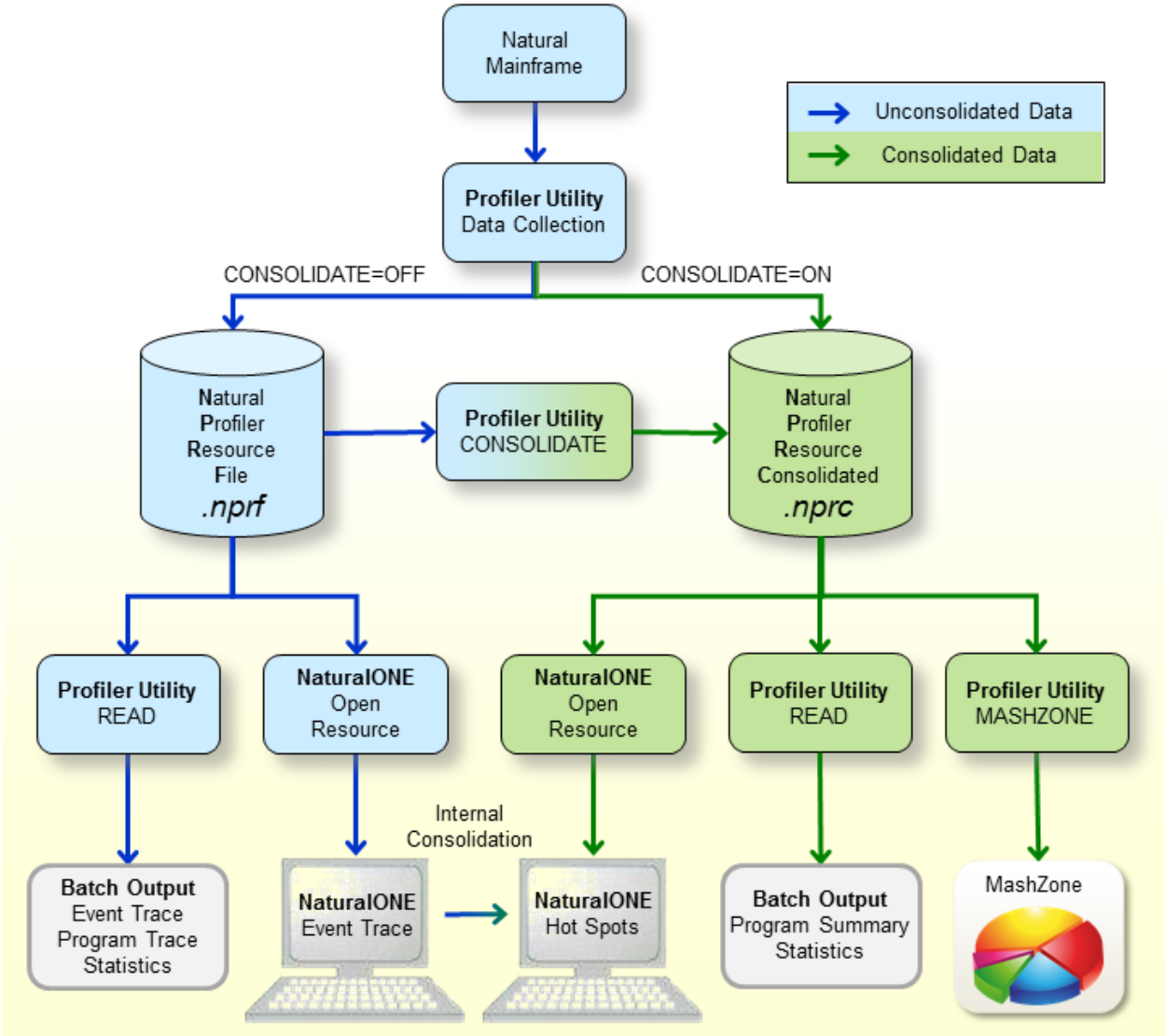
While the application is running, the GP offset of every executed statement is searched in the statement table and marked as covered. Finally, the covered statements and the not covered (those which have not been marked) statements are written to a Natural NCVF resource file.

When the NCVF coverage resource file is analyzed with the Profiler `READ` function, the source of the monitored programs is read and the lines are marked according to the coverage of the statements in the line.

**Note:** Natural code coverage is not provided for programs cataloged with the Natural Optimizer Compiler (NOC).

### Data Processing

The following graphic shows how the Profiler utility processes unconsolidated and consolidated data:




The graphic is explained in the following section:

- When a Natural mainframe batch application is profiled with the Profiler utility data collection, the resulting event data is written to a Natural Profiler resource file (NPRF) or a Natural Profiler resource consolidated (NPRC) file depending on the setting of the `CONSOLIDATE` keyword of the Profiler utility `INIT` function.
- The Natural Profiler resource file (extension `.nprf`) contains the event data in an unconsolidated format, which means that there is one record for each event.
- The Profiler utility `READ` function reads the event data from the NPRF resource file. It provides an event trace, a program trace and the Profiler statistics. The resulting data can be exported to a file in text or CSV (comma-separated values) format.
- If the NPRF resource file is opened from NaturalONE, the unconsolidated event data is listed on the NaturalONE **Event Trace** page.

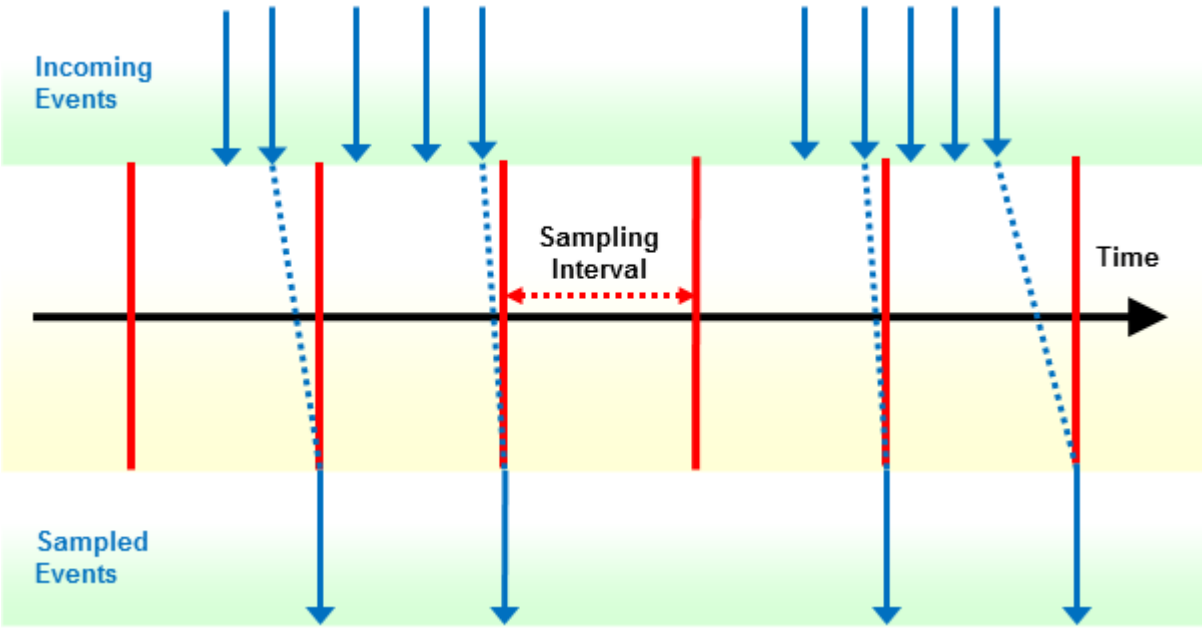
- The NaturalONE **Hot Spots** page shows the event data in a consolidated form. If the data derives from an NPRF resource file, NaturalONE consolidates the data internally.
- The Profiler utility `CONSOLIDATE` function reads the event data from the NPRF resource file, consolidates it and writes it to an NPRC resource file.
- The Natural Profiler resource consolidated file (extension `.nprc`) contains the event data in a consolidated format, which means that similar records are aggregated in one consolidated record. In general, an NPRC resource file is much smaller than the corresponding NPRF resource file and, therefore, much quicker to process.
- If the NPRC resource file is opened from NaturalONE, the consolidated event data is shown on the **Hot Spots** page. It is not possible to view the event trace because the NPRC resource file does not contain the data of each single event.
- The Profiler utility `READ` function reads the event data from the NPRC resource file. It provides a trace of the consolidated records, a program summary and the Profiler statistics. The resulting data can be exported to a file in text or CSV (comma-separated values) format.
- The Profiler utility `MASHZONE` function reads the event data from the NPRC resource file and exports it in CSV (comma-separated values) format as expected by the **Natural Profiler MashApp**.
- The **Natural Profiler MashApp** visualizes the Profiler event data and statistics in MashZone.

## Sampling

In general, profilers are classified into event-based or statistical profilers. Statistical profilers, which operate by sampling, interrupt the operating system at regular intervals to receive the profiling data. The resulting data is not exact but a statistical approximation.

The Natural Profiler is an event-based profiler. It receives control and collects the profiling data whenever a Natural event occurs. Although the Natural Profiler does not interrupt the operating system, it offers a sampling technique that generates the same profiling data as statistical profilers.

Natural Profiler sampling works like a filter: it eliminates all events except the last one in a sampling interval. Additionally, it replaces the event CPU timestamp by the subsequent sampling time. This way, the Natural Profiler only collects those events that were active at the beginning of a sampling interval.




If you use Profiler sampling, consider the following:

- Natural Profiler sampling provides a good estimation of the consumed CPU time. It does not provide other estimations such as hit counts, elapsed times, and Adabas times.
- Natural Profiler sampling is a statistical approach which reduces the number of events severely with nearly the same CPU-time results.
- The smaller the sampling interval, the more accurate the result.
- The higher the sampling interval, the less data is produced.
- The resulting event duration is a multiple of the sampling interval.
- The sampling generates at most one record per sampling interval.
- Events which spent more time than a sampling interval need one record only.
- The session termination (`ST`) event is recorded unchanged.

If the total application CPU time is known and sampling is used, the number of events can be estimated:

$$\text{Number of events} \approx \frac{\text{Total CPU time in microseconds}}{\text{Sampling interval}}$$

**Example**

In the following example application, the program `XPROF` calls three subprograms. The application is profiled twice:

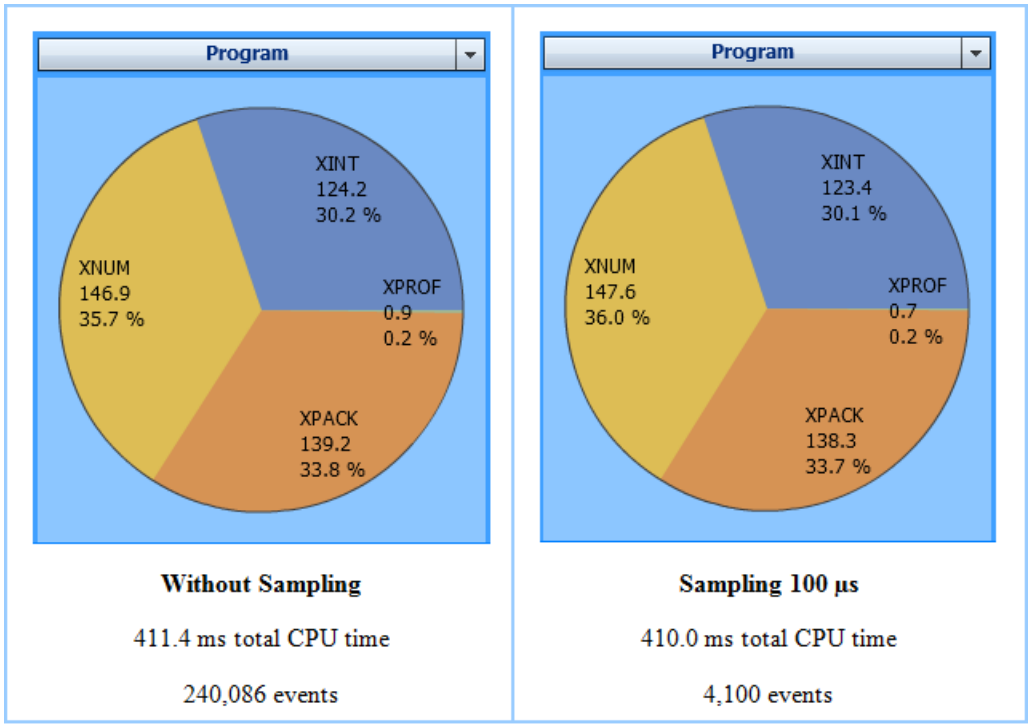
1. Without sampling.

2. With sampling whereby a sampling interval of 100 microseconds is used.

For sampling, the following keywords are specified with the Profiler utility `INIT` function:

```
FUNCTION=INIT            /* Initialize Profiling
  SAMPLING=ON            /* Use sampling
  INTERVAL=100           /* Microseconds
```

The Natural Profiler MashZone pie charts below show for each program the name of the program, the CPU time spent (in units of milliseconds) and the CPU time percentage with respect to the total CPU time. The left chart reflects the run without sampling and the right chart the run with sampling. Although the number of events has been reduced by the sampling to about 1.7 percent, the resulting CPU time and distribution are nearly the same.




## Profiler Performance in Batch

Profiling an application usually impacts the performance of the application. The impact can concern the measurement and the overall job duration. The Natural Profiler has implemented several features to keep the measurement as accurate and the performance loss as low as possible.

- The Profiler monitor session, which reads the data from the Profiler data pool and writes it into the Profiler resource file, is running as an asynchronous task. The time spent for the monitor session is therefore not taken into account for the application time measurement. Nevertheless,

because the Profiler trace session has to wait at the end until the monitor session has finished, it can impact the overall job duration.

- The NATRDC1 exit running in the Profiler trace session measures the CPU time needed for the assembling of the trace record separately and subtracts it from the session CPU time.
- If Natural statement events are collected with the event filter setting `STATEMENT=ON`, the Profiler disables the statement event generation in the Natural nucleus as long as any block filter (library, program, line, FNAT, event count or time filter) is active. This reduces the load on the Natural nucleus, the SYSRDC data collector and finally on NATRDC1 which would otherwise reject the event.
- The Profiler compresses the event data before it writes it into the resource. Compression can save up to 80 percent of data storage which reduces the number of I/Os dramatically. The event data is also read by NaturalONE in the compressed format which increases the transport flow rate.
- When running on a z/OS machine with zIIP (IBM System z Integrated Information Processor), time is lost if execution switches from the general purpose processor to the zIIP and vice versa. If you use Natural zIIP with the Profiler, the NATRDC1 exit will run on the zIIP with a minimal number of switches.

**Profiler Data Pool Size Selection**

If the Profiler data pool is full, the Profiler trace session waits one second so that the Profiler monitor session can read and release some space. If the data pool is too small, it can happen that the Profiler monitor session reads all data before the trace session is restarted. If the monitor session does not find data in the data pool, it waits one second for new data. Now both sessions wait alternating, which increases the overall job duration severely.

The **Data pool empty after full** property in the **Trace Session** category of the Profiler statistics indicates such alternating wait situations. If the value of this property is greater than zero (`0`), increase the Natural profile parameter `PDPSIZE` to an appropriate value.

**Example**

```
PDPSIZE=50000
```

## Profiling Long-Running Applications

Profiling a long-running batch application can produce a huge amount of data, especially when Natural statements are monitored.

This section describes how to minimize the number of events to be monitored while keeping essential information:

- Start and Pause Profiling

- Set Filters
- Use Sampling for CPU Analysis
- Use Server-Side Data Consolidation

### Start and Pause Profiling

- If the batch job starts multiple Natural applications then initialize and start the Profiler immediately before the first application of interest. As soon as the Profiler is initialized, it has an impact on the performance even if no events are collected.
- Pause the Profiler for applications which are not of interest and restart it for the next application of interest.
- Eventually, use the application programming interface (API) to start and pause profiling at specific points in the application.

**Example**

A job executes three Natural applications. From these three applications, only the second one is of interest for a Profiler analysis.

Initialize and start profiling immediately before the second application starts executing, and pause profiling right after execution as in the example below:

```
APP-01
PROFILER
FUNCTION=INIT,...          /* Initialize profiling
FUNCTION=START             /* Start data collection
END-PROFILER               /* End Profiler input
APP-02
PROFILER
FUNCTION=PAUSE             /* Pause data collection
END-PROFILER               /* End Profiler input
APP-03
FIN
/*
```

This way, profiling has no impact on the performance of the other applications.

### Set Filters

- Use the event filter `FNAT=OFF` to avoid monitoring Natural system programs or do not specify the `FNAT` keyword at all.
- Statement events have the most impact on the performance and quantity. The other events have only a low impact on the performance but enlarge the quantity. Monitor statement events only if you really need them. Monitor from the non-statement events only those you want to analyze.

  For example, if you want to view in NaturalONE the program hot spots but neither the statement nor the line hot spots, the following Profiler event filter setting is sufficient:

  ```
  FILTER=EVENT            /* Set event filter
    EVENT=P               /* Program events
  ```

  With this setting, only the program and session events needed for the program hot spots are monitored, whereas statement and FNAT collection are deactivated by default.
- Monitor only the libraries and programs that you want to analyze. Use program filter to restrict profiling.

  For example, if a (first) Profiler run without statement collection has shown that the most CPU time was spent in the program `HIGHCPU`, then you might only want to know in which line of this program the most time was spent and which other events (database calls, external program calls, etc.) are performed:

  ```
  FILTER=PROGRAM          /* Set program filter
    LIBRARY=PRFDEMO       /* Monitored library
    PROGRAM=HIGHCPU       /* Monitored program
  FILTER=EVENT            /* Set event filter
    EVENT=ALL             /* All events
    STATEMENT=ON          /* Collect statements (no count)
  ```

### Use Sampling for CPU Analysis

For the CPU analysis of a long-running application, we recommend **sampling**. If you use already filter settings to reduce the number of events, you can additionally activate sampling to reduce the number of events further.

Most event data is generated when statements are collected. Therefore, sampling will often be used in conjunction with statement collection. For very long-running applications, however, it might be helpful to use sampling even if no statements are collected. If you use sampling without statement collection, we recommend a sampling interval that is higher than that specified when statements are collected.

Sampling has only restricted impact on the Profiler performance but it can reduce the amount of data dramatically. The formula in the section ***Sampling*** rearranged here can be used to choose a sampling interval so that the number of events is equal to or less than an approximate value:

$$\text{Sampling interval} \geq \frac{\text{Total CPU time in microseconds}}{\text{Approximate number of events}}$$

For example, a batch application requires 40 minutes of CPU time (2,400,000,000 µs). Sampling should restrict the number of events to at most 500,000 events. The corresponding sampling interval can be calculated with the formula above.

$$\text{Sampling interval} \geq \frac{2{,}400{,}000{,}000}{500{,}000} = 4{,}800$$

Specify the following sampling setting for the Profiler utility `INIT` function:

```
SAMPLING=ON
INTERVAL=4800
```

### Use Server-Side Data Consolidation

If you want to analyze the performance of the event data and do not require an event or program trace, we recommend that you consolidate the event data on the server side. The Profiler data consolidation combines similar records into one consolidated record containing aggregated time values and a hit counter.

The event data can be consolidated during data collection with the `CONSOLIDATE` keyword of the Profiler utility `INIT` function as described in the section ***Initializing Profiling***.

Unconsolidated event data of an NPRF (Natural Profiler resource file) resource file can be consolidated with the Profiler utility `CONSOLIDATE` function as described in the section ***Consolidating Event Data***.

Consolidated data is written to an NPRC (Natural Profiler resource consolidated) resource file which is in general significantly smaller than the corresponding NPRF resource file. It opens much faster from NaturalONE and provides the same hot spots as the NPRF resource file.

**Note:** Natural code coverage data written to an NCVF resource file is automatically consolidated by Natural code coverage.

## Related Topics

- The `RDC` profile parameter configures the Natural Data Collector used by the Profiler utility and the SYSRDC utility: see *RDC - Configure the Natural Data Collector* in the *Parameter Reference* documentation.

  The CMRDC interface controls the data recorded in the Natural Data Collector buffer: see *Calling the CMRDC Interface* in the section *SYSRDC Utility* in the *Utilities* documentation.
- The use of the Profiler utility can be controlled by Natural Security, see *Protecting Utilities* in the *Natural Security* documentation.
- The use of the NaturalONE Profiler and NaturalONE code coverage is described in the *NaturalONE* documentation.

# 104 Using the Profiler Utility in Online Mode


- Prerequisites ..... 798
- Invoking and Terminating the Profiler Utility Online ..... 798
- Events ..... 799
- Functions ..... 800

The Profiler utility in online mode is designed for getting a quick view to the last actions of a Natural online application. The utility is menu-based and provides functions to control profiler tracing, to select required event types, to maintain and display trace records, to download the event data to the PC and to save the event data as Profiler resource file.

The Profiler utility helps you analyze the logical flow of Natural applications and trace the utilization of resources.

## Prerequisites

To use the Profiler utility in online mode, the keyword subparameter `SIZE` of the profile parameter `RDC` must be set to a value greater than 2, see *RDC - Configure the Natural Data Collector* in the *Parameter Reference* documentation.

In addition, the `RDC` parameter controls the following default behavior of the Profiler utility:

- By setting `RDC=ON`, profiler tracing is already active when the Profiler utility is started. By default, `RDC` is set to `OFF` and profiler tracing must be activated separately, see ***Start/Stop Profiler Tracing***.
- The keyword subparameter `EVENT` determines which event types are selected for profiler tracing. By default, `EVENT` is set to `ALL`, which covers all available event types (see *Data-Collecting Events* in the *SYSRDC Utility* documentation).
- For event types covered by the Profiler utility, see ***Events***.

## Invoking and Terminating the Profiler Utility Online

This section covers instructions for invoking and terminating the Profiler utility in online mode.

### To invoke the Profiler utility

- Enter the following system command:

  ```
  PROFILER
  ```

  A menu similar to the example below appears. In the header of the menu, the current state with respect to profiler tracing is displayed (**Trace started**/**Trace stopped**). If profiler tracing is stopped, the number of trace records collected in the Natural Data Collector buffer is also displayed.

```
12:30:57              ***** NATURAL PROFILER UTILITY *****          2016-03-07
User SAG                          - Main Menu -                    1035 records
                                                                  Trace stopped

                      Code  Function

                       S    Select Profiler events
                       L    List trace records
                       D    Display trace record
                       T    Start/Stop Profiler tracing
                       P    Print trace records
                       W    Download trace records
                       B    Save data as resource
                       ?    Help
                       .    Exit

               Code .. _    Record from .. 1_____  to .. 1035__



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

### To terminate the Profiler utility

- Press PF3 or PF12.

  Or:

  Enter a period (.) in the **Code** field.

  Or:

  Enter `EXIT` in the command line.

## Events

During a Natural session, different kinds of events can occur, for example, a program start. Data specific to an event can be collected in a trace record. Each event is associated with an event type, that is, a one- or two-letter code. The following events and event types are available:

| Event | Event Type | When the Event Occurs |
|---|---|---|
| **P**rogram **L**oad | PL | When a program (Natural object) is loaded or when it is already located in the buffer pool. |
| **P**rogram **S**tart | PS | When a program (Natural object) is started. |
| **P**rogram **T**ermination | PT | When a program (Natural object) is terminated. |
| **B**efore **D**atabase Call | DB | Before a database call is executed. |
| **A**fter **D**atabase Call | DA | After a database call has been executed. |
| **B**efore Terminal **I**/O | IB | Before a terminal input/output is executed. |
| **A**fter Terminal **I**/O | IA | After a terminal input/output has been executed. |
| **B**efore External Program Call | CB | Before an external program call (CALL statement) is executed. |
| **A**fter External Program Call | CA | After an external program call (CALL statement) has been executed. |
| Runtime **E**rror | E | When a Natural runtime error has occurred. |
| **N**atural **S**tatement | NS | When a Natural statement is executed. For technical reasons, there is no one-to-one relationship between a Natural source code statement and an object code in the cataloged object. Therefore, multiple Natural statements can be merged into one `NS` event and conversely, one Natural statement can cover multiple `NS` events. |
| **U**ser-Defined Event | U | When a user-defined event is generated using the Natural statement `CALL 'CMRDC' 'U'` (see *User-defined Events* in the *SYSRDC Utility* documentation). The first byte of the user data is interpreted as subtype. Therefore, a two-letter code is displayed for a user-defined event when the trace records are listed. |

## Functions

The functions covered by the Profiler utility in online mode can be divided into three categories:

- Function *Select Profiler Events* to select events.
- The display functions *List Trace Records*, *Display Trace Record*, *Print Trace Records*, *Download Trace Records* and *Save Data as Resource* to list, display, print, download or save trace records.
- Function *Start/Stop Profiler Tracing* to start or stop profiler tracing.

All functions provided by the Profiler utility can be invoked from the **main menu**.

### To invoke a function

- Enter a function code in the **Code** field.

  **Note:** The input fields **Record from** and **Record to** can be modified to define a range of records that is listed or displayed at first by the functions *List Trace Records* and

*Display Trace Record*, or to filter records for the functions *Print Trace Records* and *Download Trace Records*.

This section covers the following functions:

- Select Profiler Events
- List Trace Records
- Display Trace Record
- Start/Stop Profiler Tracing
- Print Trace Records
- Download Trace Records
- Save Data as Resource

## Select Profiler Events

This function is used to select event types for profiler tracing (in column **Collect**) and as filter for the display functions (in column **Display**).

### To select an event type for profiler tracing

- Enter any key in the corresponding input field in the **Collect** column. You need to select at least one event type.

  As a result, the Natural Data Collector is updated dynamically according to your selection. If profiler tracing is active, it is continued with the selections that you have made.

**Notes:**

1. To activate profiler tracing for the event type **Natural statement** (**NS**), the profile parameter `TRACE` must be set to `NATPROX` and `ITRACE` to `ON`, see the *Parameter Reference* documentation. If you do not supply these values before starting your Natural session, you are prompted to do so when you select this event type.
2. At session start, the event types selected for profiler tracing are specified by the `RDC` subparameter `EVENT`; see *EVENT - Natural Data Collector Events to be Recorded* in the *Parameter Reference* documentation. Any modification of this default selection is valid for the whole Natural session.

### To select an event type as filter for display functions

- Enter any key in the corresponding input field in the **Display** column. You need to select at least one event type.

**Note:** At session start, the event types selected for display functions are specified by the `RDC` subparameter `EVENT` (see above). Any modification of this default selection is valid until

the next LOGON. After this, the set of event types selected is equal to the set of event types selected for profiler tracing (see above).

### PF Keys

The following PF keys are available:

| PF Key | Name | Function |
|---|---|---|
| PF1 | Help | Display context-sensitive help. There is a specific help text for each input field. In other contexts, for example the command line, a general help text is displayed. |
| PF3 | Exit | Exit the current menu. |
| PF8 | Reset | Unselect all event types. |
| PF9 | All | Select all event types. |
| PF12 | Canc | Exit the current menu. |

### Example

In the example below, the following event types are selected for profiler tracing: PL, PS, PT, DB, DA, CB, CA, NS and U. The event types PL, PS, PT, DA, CB, CA and U are selected as filter for display functions.

```
10:13:24              ***** NATURAL PROFILER UTILITY *****          2011-03-29
User SAG                   - Select Profiler Events -               79 records
                                                                 Trace stopped

              Collect Display Type Profiler Event
              ------- ------- ---- ----------------------------
                 X       X     PL  Program load
                 X       X     PS  Program start
                 X       X     PT  Program termination
                 X       _     DB  Before database call
                 X       X     DA  After database call
                 _       _     IB  Before terminal I/O
                 _       _     IA  After terminal I/O
                 X       X     CB  Before external program call
                 X       X     CA  After external program call
                 _       _     E   Runtime error
                 X       _     NS  Natural statement
                 X       X     U   User-defined event



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

## List Trace Records

This function is used to list all trace records collected in the Natural Data Collector buffer. For this purpose, profiler tracing is stopped. A trace record can only be shown if the corresponding event type has been selected for the display functions by the *Select Profiler Events* function.

When the function is invoked, the trace record displayed at first is specified by the **Record from** field, see **main menu**. In addition, this trace record is also marked by an `X`, and the trace record specified by the **Record to** field by a `Y`.

This section covers the following topics:

- Structure of Trace Records Displayed
- Navigation
- Updating Event Types to be Displayed
- Line Commands
- Local Commands
- PF Keys
- Example

### Structure of Trace Records Displayed

Each trace record is displayed on a separate line which is headed by an input field for line commands (**C**). The record itself contains general and event-specific data. General data comprises features common to all records such as record number (**Record**), event type (**Ev**) or event time (**Time**). Event-specific data comprises features specific to a view, that is, a group of related events. For example, the event types `PL`, `PS` and `PT` constitute the view `Program Load/Start/Termination`. The view `General Information` includes all event types and covers the general data. The table below gives an overview on the available views and the event types associated:

| View | Code | Event Types |
|---|---|---|
| **G**eneral Information | G | all |
| **P**rogram Load/Start/Termination | P | PL, PS, PT |
| Before/After **D**atabase Call | D | DB, DA |
| Before/After Terminal **I**/O | I | IB, IA |
| Before/After External Program **C**all | C | CB, CA |
| Runtime **E**rror | E | E |
| **N**atural **S**tatement | N | NS |
| **U**ser-Defined Event | U | U |

**Notes:**

1. General Information (G) is the current view by default.

2. If there is a view for which no event type has been selected to be displayed, the letter code for that view is replaced by a minus sign (**-**).

### Navigation

The list of trace records displayed by the *List Trace Records* function can be quite large with respect to the size of the records displayed and the number of records listed. This section covers the following options:

- Navigation within a Record
- Navigation within a List of Records

### Navigation within a Record

To navigate within a record, you need to change the current view.

#### To change the current view

- Select a letter code in **View** using the cursor and press ENTER. See the **table above** for available letter codes.

  Or:

  Press PF10 (**<**) to choose the neighboring view on the left. Press PF11 (**>**) to choose the neighboring view on the right.

### Navigation within a List of Records

#### To navigate within a list of records

- Enter a record number in the input field **Record** to position to the corresponding trace record. Note that the trace record selected must have an event type that has been selected for the display functions by the *Select Profiler Events* function.

  Or:

  Enter an event type in the input field **Ev** to step forward to the next trace record marked with such an event type. From this position, continue to press ENTER to step to the next matching occurrence. Use asterisk notation (*) to match a group of event types, for example `D*` for all database calls.

  Or:

  Press PF6 (**--**), PF7 (**-**), PF8 (**+**) or PF9 (**++**) to scroll through the list.

### Updating Event Types to be Displayed

A selection of event types for display functions can be updated by pressing PF4 (**DiEv**). This opens a menu similar to the menu which is opened when choosing the *Select Profiler Events* function, with the exception that event types can only be selected for display functions. See ***PF Keys*** in *Select Profiler Events* for available PF keys.

### Line Commands

You can enter a line command in the **C** column next to the trace record you have selected. The following line commands are available:

| Local Command | Function |
|---|---|
| D | Display all information available for the selected record. See also ***Display Trace Records***. |
| X | Mark the line with an **X**. The **Record from** field in the **main menu** is updated accordingly. |
| Y | Mark the line with a **Y**. The **Record to** field in the **main menu** is updated accordingly. |

### Local Commands

You can enter a local command in the **Command** line. The following local commands are available:

| Local Command | Function |
|---|---|
| X | Scroll to the line marked by an **X** (and indicated by **Record from**). |
| Y | Scroll to the line marked by a **Y** (and indicated by **Record to**). |

### PF Keys

The following PF keys are available:

| PF Key | Name | Function |
|---|---|---|
| PF1 | Help | Display context-sensitive help. See **above**. |
| PF2 | Disp | Display all information available for the selected trace record. See also ***Display Trace Records***. |
| PF3 | Exit | Exit the current menu. |
| PF4 | DiEv | Modify the selection of event types to be displayed. See section ***Updating Event Types to be Displayed***. |
| PF5 | Hex | Display user data in hexadecimal mode (only for event type **U**). |
| | Struc | Display program structure (only for event types **PL**, **PS** and **PT**). |
| PF6 | -- | Scroll to the beginning of the list. |
| PF7 | - | Scroll one page up. |
| PF8 | + | Scroll one page down. |
| PF9 | ++ | Scroll to the end of the list. |

| PF Key | Name | Function |
|---|---|---|
| PF10 | < | Select the view on the left side of the current view. |
| PF11 | > | Select the view on the right side of the current view. |
| PF12 | Canc | Exit the current menu. |

### Example

The following menu lists trace records for event types PL, PS, PT and DA, which have been selected in the example **above**, and displays trace information for view P which covers the event-specific data of the event types PL, PS and PT.

```
10:13:39                ***** NATURAL PROFILER UTILITY *****          2011-03-29
User SAG                      - List Trace Records -                  79 records
                        View:  G P D - C - - U
                               PL/PS/PT - Program Load/Start/Termination

  C Record Ev Library  Program  Type DBID  FNR
  - ______ __ -------- -------- ---- ----- -----
X _ 000001 PL SYSEXPG  DYNAMX06
  _ 000002 PS SYSEXPG  DYNAMX06  P      10  2430
  _ 000010 PL SYSEXPG  DYNAMX05
  _ 000011 PS SYSEXPG  DYNAMX05  N      10  2430
  _ 000018 PT SYSEXPG  DYNAMX05  N      10  2430
  _ 000025 PT SYSEXPG  DYNAMX06  P      10  2430
  _ 000026 PL SYSEXPG  EDITMX02
  _ 000027 PS SYSEXPG  EDITMX02  P      10  2430
  _ 000031 DA
  _ 000040 DA
  _ 000047 DA
  _ 000054 DA
  _ 000057 PT SYSEXPG  EDITMX02  P      10  2430
Use PF10/PF11 to show event-specific data of the trace records.
Command ===>
Enter-PF1---PF2---PF3---PF4---PF5---PF6---PF7---PF8---PF9---PF10--PF11--PF12---
      Help  Disp  Exit  DiEv  Struc --    -     +     ++    <     >     Canc
```

### Display Trace Record

This function is used to display all tracing information for each trace record separately. For this purpose, profiler tracing is stopped. When the function is invoked, the trace record displayed at first is specified by the **Record from** field, see **main menu**.

This section covers the following topics:

- Navigation
- PF Keys

- Example

### Navigation

#### To navigate to another trace record

- Enter a record number in the input field **Record** to position to the corresponding trace record. Note that the trace record selected must have an event type that has been selected for the display functions by the *Select Profiler Events* function.

  Or:

  Press PF6 (**--**), PF7 (**-**), PF8 (**+**) or PF9 (**++**) to scroll through the sequence of trace records displayed.

### PF Keys

The following PF keys are available:

| PF Key | Name | Function |
|---|---|---|
| PF1 | Help | Display context-sensitive help. See **above**. |
| PF3 | Exit | Exit the current menu. |
| PF5 | Hex | Display user data in hexadecimal mode (only for event type **U**). |
| PF6 | -- | Display the first trace record. |
| PF7 | - | Display the preceding trace record. |
| PF8 | + | Display the succeeding trace record. |
| PF9 | ++ | Display the last trace record. |
| PF10 | LongE | Show long error message (only defined for event type **E**). |
| PF12 | Canc | Exit the current menu. |

### Example

In the example below, trace record `10` which has already been listed in the example **above** is displayed with all trace information.

```
10:14:01              ***** NATURAL PROFILER UTILITY *****          2011-03-29
User SAG                    - Display Trace Record -                79 records


Record / Event .... 10____  /  PL - Program load

Event time ........ 10:12:14.130696    Elapsed time ....... 0.000157
CPU time .......... 0
Current user ID ... SAG                Current group ID ...
Current program ... DYNAMX06           Library ............ SYSEXPG
Program level .....                    Program line ....... 0160

Loaded program .... DYNAMX05           Invocation type .... CALLNAT
Library ........... SYSEXPG





Enter required record number, or use PF6 - PF9 for paging.
Command ===>
Enter-PF1---PF2---PF3---PF4---PF5---PF6---PF7---PF8---PF9---PF10--PF11--PF12---
      Help        Exit              --    -     +     ++                Canc
```

### Start/Stop Profiler Tracing

This function is used to switch profiling on or off. If profiler tracing has been active, it is stopped. Conversely, if it has been inactive, it is started. As a result, the new state is displayed in the header of the main menu.

#### To start or stop profiler tracing

- Enter T in the **Code** field and press ENTER.

**Note:** As an alternative, you can also use the commands described in the *Trace Recording* section of the *SYSRDC Utility* documentation.

### Print Trace Records

This function is used to print the trace records within the range defined by the input fields **Record from** and **Record to**. For this purpose, profiler tracing is stopped. Note that a trace record can only be printed if the corresponding event type has been selected for the display functions by the *Select Profiler Events* function.

#### To print trace records

1 Enter P in the **Code** field.

As a result, a separate window is opened in which you can modify the (default) printer name or the range of trace records to be printed. You can also specify if the trace records are printed with header information. The printer used is identified by `print file 1`.

2 Press ENTER.

### Download Trace Records

This function is used to download the trace records within the range defined by the input fields **Record from** and **Record to**. For this purpose, profiler tracing is stopped. Note that a trace record can only be downloaded if the corresponding event type has been selected for the display functions by the *Select Profiler Events* function.

#### To download trace records

1 Enter W in the **Code** field.

As a result, a separate window is opened in which you can modify the range of trace records to be downloaded. The download file used is identified by `work file 7`. For downloading to the PC, we recommend that you use Natural Connection.

2 Press ENTER.

**Notes:**

1. Specify `.htm` for the extension of the downloaded file.
2. We recommend that you use MS Excel to display the downloaded trace records. This requires that MS Excel and Natural have compatible settings, for example with respect to the decimal character used.

### Save Data as Resource

This function is used to save all trace records in a Profiler resource file (extension `.nprf`). For this purpose, profiler tracing is stopped.

The Profiler resource file can be read by NaturalONE and by the Profiler utility in batch mode which provide a performance analysis (hot spots) and other evaluations. For more information, see the *NaturalONE* documentation and ***Using the Profiler Utility in Batch Mode***.

**Notes:**

1. If the Profiler utility runs under CICS or Com-plete, only the elapsed time is provided but not the CPU time.
2. The event data collected by the Profiler utility in online mode does not contain copycode information.

### To save the data as resource

1 Enter B in the **Code** field.

As a result, a separate window is opened in which you can modify the (default) resource name and library. You can also specify whether the resource should be replaced, if it already exists.

The default resource name is `*INIT-USER_yyyymmdd_hhiiss` where `*INIT-USER` is the user ID under which the Natural session is running and *`yyyymmdd_hhiiss`* is the current date and time. If the extension `.nprf` (Natural Profiler resource file) has not been specified with the resource name, it is added automatically.

The default library is the current library.

2 Press ENTER.

# 105 Using the Profiler Utility in Batch Mode


- Quick Start for Profiling ..... 812
- Quick Start for Code Coverage ..... 815
- Prerequisites ..... 818
- Invoking and Terminating the Profiler Utility ..... 821
- Syntax and Keywords ..... 821
- Events and Data Collected ..... 825
- Initializing Profiling ..... 830
- Initializing Code Coverage ..... 833
- Starting and Pausing Data Collection ..... 836
- Using Filters to Limit the Data Collected ..... 839
- Enabling Sampling ..... 844
- Writing User-Defined Events ..... 845
- Monitor Session CMPRMIN ..... 846
- Profiling a Batch Natural RPC Server ..... 848
- Profiling a Mainframe Session from Natural Studio ..... 849
- Consolidating Event Data ..... 851
- Evaluating Event Data ..... 853
- Exporting Event Data for MashZone ..... 871
- Maintaining Profiler Resource Files ..... 872
- Including Profiler Input from Natural Text Objects ..... 876
- Event Trace ..... 877
- Tracing Natural Code Coverage ..... 879
- Internal Trace ..... 882
- Profiler Statistics ..... 884

The Natural Profiler is used to monitor the internal process flow of a Natural batch application and to analyze the performance and the code coverage of the application.

The Profiler utility is controlled by JCL input and provides functions for data collection and data processing:

1. The **data collection** functions control Profiler tracing, select required event types, filter, consolidate (aggregate) or sample data and write the resulting events to the Profiler resource file.
2. The **data processing** functions read and process the event data from the Profiler resource file. Unconsolidated event data can be consolidated.

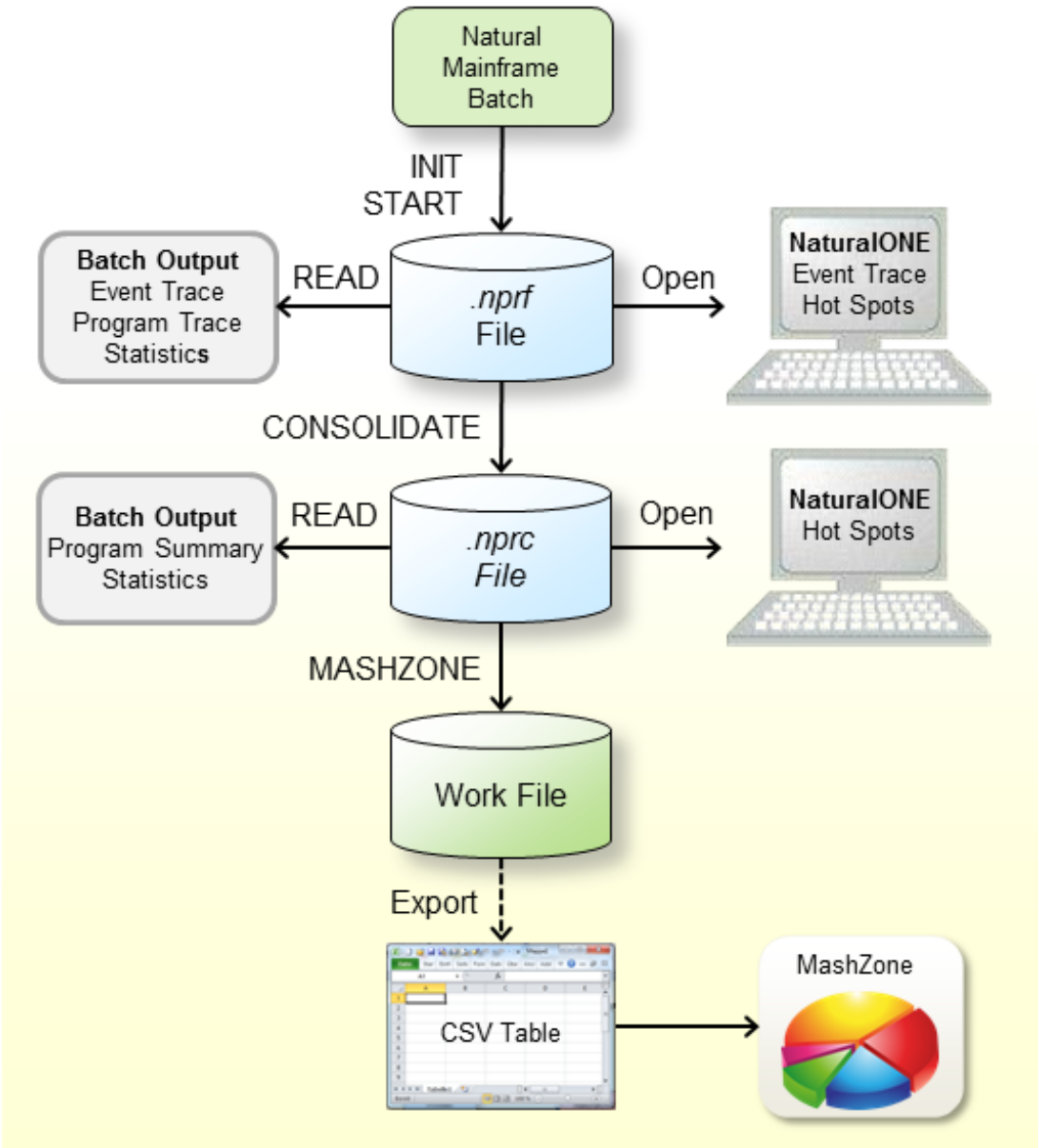
   You can output statistics, a program summary, a program trace, an event trace with the most important data, and reports on program and statement coverage. You can export the resulting data in text or CSV (comma-separated values) format.

The Profiler resource file can be read by NaturalONE which displays the full event trace and provides a performance analysis (hot spots) of the Natural batch application. Coverage data can be inspected in the NaturalONE **Coverage** view and in the NaturalONE source editor. The exported profiling event data can be analyzed with the **Natural Profiler MashApp** which visualizes the data on an interactive MashZone dashboard.

## Quick Start for Profiling

This section briefly describes the steps required for profiling Natural batch applications and viewing the results. The instructions provided here may serve as a guideline when starting to use the Natural Profiler. Detailed information regarding the steps is provided in the remainder of this chapter.

The steps to take depend on the evaluation you want to perform for your application as illustrated in the following graphic:




1. Check that the **prerequisites** are met.
2. Add the Profiler utility `INIT` and `START` functions to the Natural batch job to start the event data collection. Example for z/OS:

```
//CMSYNIN  DD  *
PROFILER
FUNCTION=INIT                /* Initialize profiling
  RESOURCE=ON                /* Write to resource
  RESOURCE-NAME='ResNam',REPLACE=YES  /* Resource name
  RESOURCE-LIB=RESLIB        /* Resource library
FUNCTION=START               /* Start data collection
END-PROFILER                 /* End Profiler input
LOGON PRFDEMO
XPROF
10000
FIN
```

In the example above, the Profiler event data is written to a resource file with the name `ResNam.nprf` in the library `RESLIB`. See also ***Initializing Profiling*** and ***Starting and Pausing Data Collection***.

3. Open the NPRF resource in NaturalONE to view the hot spots and the event trace.
4. Submit a Natural batch job with the Profiler utility `READ` function to print an event trace, a program trace and the Profiler statistics. Example:

```
FUNCTION=READ           /* Read Profiler data
  RESOURCE-LIB=RESLIB   /* Resource library
  RESOURCE-TYPE=NPRF    /* Use resource type NPRF
  EVENT=ON              /* Print event trace
  PROGRAM=ON            /* Print program trace
  STATISTICS=ON         /* Print statistics
```

See also ***Profiler Utility READ Function***.

5. Submit a Natural batch job with the Profiler utility `CONSOLIDATE` function to consolidate (aggregate) the event data. Example:

```
FUNCTION=CONSOLIDATE    /* Consolidate Profiler data
  RESOURCE-LIB=RESLIB   /* Resource library
  REPLACE=YES           /* Replace resource
```

The consolidated Profiler event data is written to the resource `ResNam.nprc` in the library `RESLIB`. See ***Consolidating Event Data***.

6. Open the NPRC resource in NaturalONE to view the hot spots.
7. Submit a Natural batch job with the Profiler utility `READ` function to generate a program summary and the Profiler statistics. Example:

```
FUNCTION=READ            /* Read Profiler Data
  RESOURCE-LIB=RESLIB    /* Resource library
  RESOURCE-TYPE=NPRC     /* Use resource type NPRC
  PROGRAM=ON             /* Print program summary
  STATISTICS=ON          /* Print statistics
```

See also *Profiler Utility READ Function*.

8. Submit a Natural batch job with the Profiler utility `MASHZONE` function to write the data to Work File 7 in the format expected by the Natural Profiler MashApp. Example:

```
FUNCTION=MASHZONE        /* Write MashZone format to Work File 7
  RESOURCE-LIB=RESLIB    /* Resource library
```

See also *Exporting Event Data for MashZone*.

9. Export the data of Work File 7 with any tool (such as FTP) as a CSV (comma-separated values) file to the Natural Profiler data directory in the MashZone environment.
10. Enter a reference to the new file in the `Overview.csv` file in the `resources\Profiler` directory.

    Open the **Natural Profiler MashApp** and select the corresponding input file to evaluate the event data.

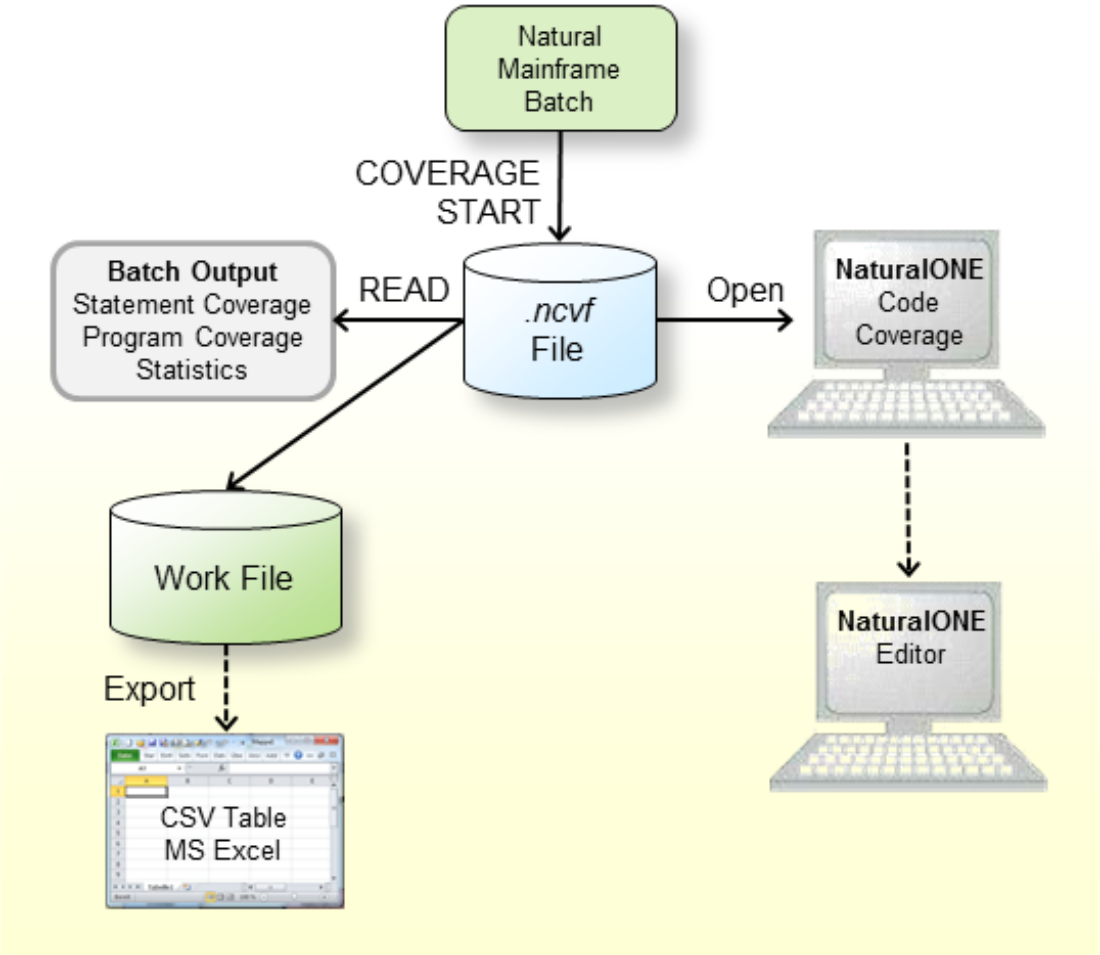
**Notes:**

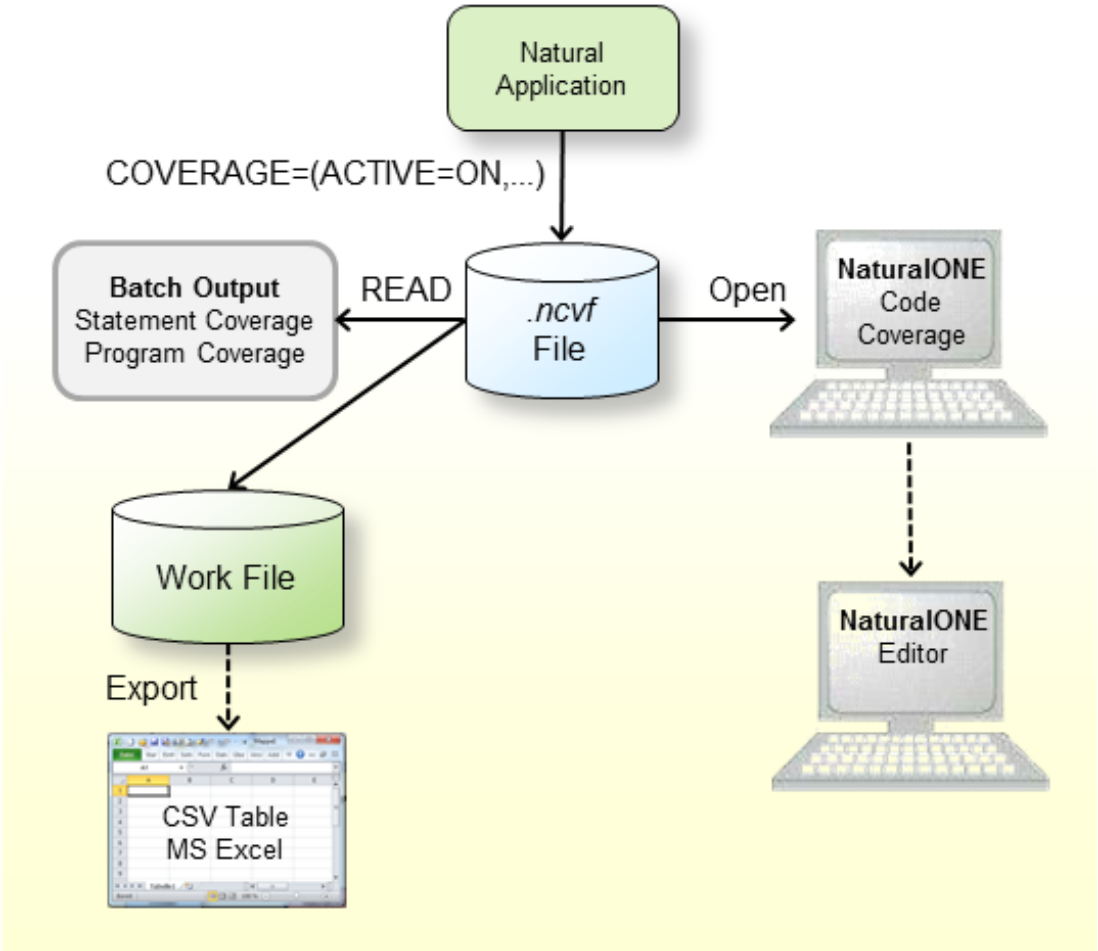
1. If the resource name is not explicitly specified in the `READ`, `CONSOLIDATE` or `MASHZONE` function of the Profiler utility, the last created NPRF or NPRC resource in the library is used.
2. If you plan to profile a long-running batch application, refer to the section *Profiling Long-Running Applications*. It covers strategies of how to minimize the number of events to be monitored.
3. The NaturalONE Profiler is described in the *NaturalONE* documentation.

## Quick Start for Code Coverage

This section briefly describes the steps required for performing the code coverage of a Natural batch applications and viewing the results. The instructions provided here may serve as a guideline when starting to use **Natural code coverage**. Detailed information regarding the steps is provided in the remainder of this chapter.

The steps to take depend on the evaluation you want to perform for your application as illustrated in the following graphic:


Natural
Mainframe
Batch
COVERAGE
START
Batch Output
Statement Coverage
Program Coverage
Statistics
READ
.ncvf
File
Open
NaturalONE
Code
Coverage
Work File
NaturalONE
Editor
Export
CSV Table
MS Excel




1. Check that the prerequisites are met.
2. Add the Profiler utility `COVERAGE` and `START` functions to the Natural batch job to start the code coverage data collection.

   Example for z/OS:

```
//CMSYNIN  DD  *
PROFILER
FUNCTION=COVERAGE            /* Initialize coverage
  RESOURCE=ON                /* Write to resource
  RESOURCE-NAME='ResNam'     /* Resource name
  REPLACE=YES                /* Replace the resource
  RESOURCE-LIB=RESLIB        /* Resource library
FUNCTION=START               /* Start data collection
END-PROFILER                 /* End Profiler input
LOGON COVDEMO
TESTCOVP
FIN
```

In the example above, the Profiler coverage data is written to a resource file with the name `ResNam.ncvf` in the library `RESLIB`. See also ***Initializing Code Coverage*** and ***Starting and Pausing Data Collection***.

3. Open the NCVF resource in NaturalONE to obtain the **Code Coverage** view.
4. From the NaturalONE **Code Coverage** view, you can directly edit the source. The editor shows all lines containing covered statements with a green background.
5. Submit a Natural batch job with the Profiler utility `READ` function to print the program and statement coverage and the Profiler statistics.

   Example:

```
FUNCTION=READ            /* Read Profiler data
  RESOURCE-LIB=RESLIB    /* Resource library
  RESOURCE-TYPE=NCVF     /* Use resource type
  EVENT=ON               /* Print statement co
  PROGRAM=ON             /* Print program cove
  STATISTICS=ON          /* Print statistics
  EXPORT=ON              /* write to work 7
  FORMAT=C               /* Semicolon/Comma/Text
```

   If the `EXPORT` keyword of the Profiler utility `READ` function is switched on, the output is written to Work File 7. If `FORMAT` is specified as `C` or `S`, the result is written as comma-separated values (CSV) where a comma or a semicolon is used as a separator, respectively.

6. Export the data of Work File 7 with any tool (such as FTP) as a CSV-formatted file to a Windows environment if you want to process it further in Microsoft Excel.

**Notes:**

1. If the resource name is not explicitly specified in the `READ` function of the Profiler utility, the NCVF resource created last in the library is used.
2. The NaturalONE **Code Coverage** view and editor are described in the *NaturalONE* documentation.

## Prerequisites

The following prerequisites must be met before you can use the Profiler utility in batch mode:

- Natural Parameter Settings
- Natural Nucleus Linkage
- Display Resource Files in Natural Development Server Environments

- Activate Data Processing

### Natural Parameter Settings

For the Natural Profiler data collection functions (`INIT`, `COVERAGE`, `START`, `PAUSE` and `TEXT`), you must set the following Natural profile parameters.

- `RDCSIZE` - Activate the Natural Data Collector (without recording data in the buffer of the Natural Data Collector):

```
RDCSIZE=2
```

- `RDCEXIT` - Define NATRDC1 as a user exit for the Natural Data Collector:

```
RDCEXIT=NATRDC1
```

- `PDPSIZE` - Optional parameter you can set additionally to determine the size of the Profiler data pool, for example:

```
PDPSIZE=1000
```

The Profiler utility data processing functions (`CONSOLIDATE`, `READ`, `MASHZONE`, `LIST` and `DELETE`) cannot be executed if profiling is active. For performance reasons, we recommend that you also deactivate the Natural Data Collector for these functions with the following (default) parameter setting:

```
RDCSIZE=0
```

For details regarding the Natural profile parameters mentioned above, see the relevant sections in the *Parameter Reference* documentation.

### Natural Nucleus Linkage

The Natural nucleus must be reentrant and reusable. Use the following linkage options:

```
RENT,REUS
```

### Display Resource Files in Natural Development Server Environments

By default, Natural mainframe resource files are not displayed in Natural Development Server (NDV) environments such as NaturalONE or Natural Studio. For analyzing the Profiler event data, the resource file must be accessed from NaturalONE. Therefore, you have to modify the NDV behavior.

**To display resource files in an NDV environment**

1 Copy the source code of the NDV user exit `NDV-SX03` from the Natural system library `SYSLIB` into a user library.

2 Edit the member. Adjust the code as described below:

```
DISPLAY-RESOURCES := 'Y'   /* Display resources in NaturalONE/Studio
```

3 Catalog it under the name `NDV-UX03`.

4 Copy it back into the system library `SYSLIB` or into the library `SYSLIBS` or `SYSTEM`.

### Activate Data Processing

If NaturalONE is installed at your site, you can activate the Profiler utility data processing functions (`CONSOLIDATE`, `READ`, `MASHZONE`, `LIST` and `DELETE`) with the following steps:

1. Start NaturalONE.
2. In the **Natural Server** view, map to the environment where the Profiler resources reside.
3. Add the program `ACTIVATE` contained in the system library `SYSPRFLR` to a new or existing project in NaturalONE.
4. Profile the program `ACTIVATE` with the context menu function **Profile As** > **Natural Application**.
5. Verify that the user-defined event data on the **Event Trace** page of the NaturalONE Profiler contains the activation success message.

When the program `ACTIVATE` is profiled, a NaturalONE Profiler key is generated and written to the Natural resource `NaturalONEProfilerKey.nprk` in the system library `SYSPRFLR`. Each Profiler data processing function reads this resource and checks the key. If the key is valid, the function is performed. A newly generated key is valid for one year. It can always be regenerated.

The Profiler data processing function starts issuing a warning 9 days before the key expires, and returns an error message if no key is found or if the key is not valid.

## Invoking and Terminating the Profiler Utility

This section provides instructions for invoking and terminating the Profiler utility in batch mode.

### To invoke the Profiler utility

- Enter the following system command into the primary command input data set `CMSYNIN`:

```
PROFILER
```

**Note:** After the `PROFILER` system command, the Profiler expects one or more lines with Profiler keyword entries.

### To terminate the Profiler utility

- Enter the following Profiler keyword into the primary command input data set `CMSYNIN`:

```
END-PROFILER
```

Or:

```
END
```

Or:

```
.
```

## Syntax and Keywords

The Profiler utility in batch mode reads the Profiler keywords that control the profiling from the primary command input data set `CMSYNIN`. The Profiler reads the input lines until it reaches the `END-PROFILER` keyword (or `END` or `.`).

This section covers the following topics:


- Profiler Utility Syntax

- Profiler Utility Keywords

## Profiler Utility Syntax

The symbols used in the syntax diagrams shown in this section are explained in *System Command Syntax* in the *System Commands* documentation.

You enter a Profiler utility command using either of the following syntax formats:

```
keyword[=value][,keyword[=value]]...
```

Or:

```
keyword
[value]
...
```

**Notes:**

1. If a value is associated with a keyword but no equal sign is found, the Profiler expects the value in a separate input line without any other keyword (second syntax format).
2. The first syntax format expects input in delimiter mode (`IM=D`).
3. The second syntax format can be used if the Profiler is to be executed with the Natural `STACK` profile parameter or if the data is entered in forms mode (`IM=F`).

The following rules apply:

- Empty lines and lines starting with an asterisk (`*`) are ignored.
- All characters in a line from `/*` to `*/` or to the end of the line are ignored.
- Some keywords have no associated value.
- Blanks can be added before or after the keyword or value.
- Multiple keywords in a line are separated by commas (applies to the first syntax format only).
- A value can be enclosed in apostrophes (`'value'`).
- A value must not contain a comma.
- Keywords and values can be specified in upper or lower case.
- The maximum input line length is 78 characters.

The Profiler utility can be executed multiple times in one Natural session. For example, it is first executed with the `INIT` and `START` functions, and then, after the execution of a user program, it is executed with the `PAUSE` function.

**Example**

The following Natural batch example (on z/OS) shows the original JCL which runs the `XPROF` program on the `PRFDEMO` library (lines in normal font) and the Profiler utility input lines which are used to profile the `XPROF` program (lines in bold).

```
//CMSYNIN  DD  *
PROFILER
FUNCTION=INIT                /* Initialize profiling
  RESOURCE=ON                /* Write to resource
  RESOURCE-NAME='Demo01',REPLACE=YES  /* Resource name
  RESOURCE-LIB=PRFDATA       /* Resource library
FUNCTION=START               /* Start data collection
END-PROFILER                 /* End Profiler input
LOGON PRFDEMO
XPROF
10000
FIN
```

The following Natural batch example (on z/OS) demonstrates how the Profiler is to be executed with the Natural `STACK` profile parameter.

```
STACK=(
 PROFILER FUNCTION:INIT:
   RESOURCE:ON:
   RESOURCE-LIB:PRFDATA:
   REPLACE:YES:
 FUNCTION:START:
 END-PROFILER;
 LOGON PRFDEMO
 )
```

### Profiler Utility Keywords

The main keywords used in the syntax of the Profiler utility in batch mode are described in the following table. Any additional (subordinate) keywords available for a main keyword are described in the sections referenced in the table. In general, a subordinate keyword value must follow the main keyword value, for example:

```
FUNCTION=READ
  PRINT=ON
```

A subordinate keyword specified before the first `FUNCTION` or `FILTER` keyword is treated as a subordinate keyword of the first `FUNCTION` or `FILTER` keyword.

The following main keywords are available:

| Keyword | Value | Description |
|---|---|---|
| FUNCTION | | Perform a Profiler utility function. |
| | CONSOLIDATE | Consolidate (aggregate) resource data. See *Consolidating Event Data*. |
| | COVERAGE | Initialize Natural code coverage. This function is mandatory for the code coverage data collection. See *Initializing Code Coverage*. |
| | DELETE | Delete a Natural Profiler resource file. See *Maintaining Profiler Resource Files*. |
| | INIT | Initialize profiling. This function is mandatory for the profiling data collection. See *Initializing Profiling*. |
| | LIST | List Profiler resources. See *Listing Profiler Resource Files* in *Maintaining Profiler Resource Files*. |
| | MASHZONE | Export resource data in MashZone format. See *Exporting Event Data for MashZone*. |
| | PAUSE | Pause the data collection. See *Starting and Pausing Data Collection*. |
| | READ | Read and evaluate resource data. See *Evaluating Event Data*. |
| | START | Start or restart the data collection. See *Starting and Pausing Data Collection*. |
| | TEXT | Add a user event to the event data. See *Writing User-Defined Events*. |
| FILTER | | Define Profiler filters to reduce the amount of event data. See *Using Filters to Limit the Data Collected*. |
| | COUNT | Set the event count filter. |
| | EVENT | Set the event, FNAT and statement filters. |
| | PROGRAM | Set the library, program and line filters. |
| | TIME | Set the CPU time filter. |
| RPC | | Profile a batch Natural RPC server. See *Profiling a Batch Natural RPC Server*. |
| ON-ERROR | | Determine how Profiler error situations are handled.<br><br>Default: TERMINATE |
| | CONTINUE | The profiling is stopped but the Natural session continues. |
| | TERMINATE | The Natural Profiler forces a termination of the Natural session. |
| TRACE | 0 - 10 | Set the level of internal trace of the Profiler trace session. The internal trace contains information such as Profiler errors and is written to the standard output of the trace session (CMPRINT data set). See *Internal Trace*.<br><br>Default: 2 (warning) |
| HELP | | A summarized description of the Profiler keywords is written to standard output. |
| INCLUDE | *object-name* | The name of the Natural text object that contains Profiler input data.<br><br>See also *Including Profiler Input from Natural Text Objects*. |

| Keyword | Value | Description |
|---|---|---|
| INCLUDE-LIB | *library-name* | The name of the Natural library that contains the text object specified with the INCLUDE keyword.<br><br>If the Natural system variable *LIBRARY-ID is specified, the name of the current library is used.<br><br>The library name is used for all following INCLUDE keywords.<br><br>Default: If INCLUDE-LIB is not specified before an INCLUDE keyword, the Natural system library SYSPRFLR is used by default.<br><br>See also ***Including Profiler Input from Natural Text Objects***. |
| END-PROFILER<br><br>or<br><br>END<br><br>or<br><br>. | | End of Profiler input. The keyword END-PROFILER, END or a period (.) indicates the end of the Profiler input. |
| ONLINE | | Perform the online Profiler utility in a batch run. This option is for compatibility with earlier Natural versions, when the Profiler utility in batch mode started the Profiler online menu. If this functionality is still required, use ONLINE as the first keyword. All subsequent input is handled by the Profiler online menus. |

## Events and Data Collected

This section describes the events and data processed by the Profiler utility in batch mode.

- Events
- Data Collected

### Events

During a Natural session, different types of events can occur (for example, a program start) where the Profiler collects data specific to the event in a trace record. Each event is associated with an event type, that is, a one or two letter code. Related event types are combined into an event group which is denoted by a one letter code.

The following events, event types and event groups are available:

| Event | Event Type | Event Group | When the Event Occurs |
|---|---|---|---|
| **S**ession **I**nitialization | SI | S | When a Natural batch session is initialized. Because the Profiler monitor session starts after the trace session, this event cannot be monitored. |
| **S**ession **T**ermination | ST | S | When a Natural batch session is terminated. The Profiler always monitors this event. |
| **P**rogram **L**oad | PL | P | When a program (Natural object) is loaded or when it is already located in the buffer pool. |
| **P**rogram **S**tart | PS | P | When a program (Natural object) is started. |
| **P**rogram **T**ermination | PT | P | When a program (Natural object) is terminated. |
| **P**rogram **R**esume | PR | P | When a program (Natural object) resumes control after another Natural object has been executed or when control returns to level 0 (no program active). |
| **P**rogram **I**nformation | PI | P | When a program (Natural object) is accessed for the first time. This event is only triggered at Natural code coverage. |
| **B**efore **D**atabase Call | DB | D | Before a database call is executed. |
| **A**fter **D**atabase Call | DA | D | After a database call has been executed. |
| **B**efore Terminal **I**/O | IB | I | Before a terminal input/output is executed. |
| **A**fter Terminal **I**/O | IA | I | After a terminal input/output has been executed. |
| **B**efore External Program **C**all | CB | C | Before an external program call (CALL statement) is executed. |
| **A**fter External Program **C**all | CA | C | After an external program call (CALL statement) has been executed. |
| Runtime **E**rror | E | E | When a Natural runtime error has occurred. |
| **N**atural **S**tatement | NS | N | When a Natural statement is executed. For technical reasons, there is no one-to-one relationship between a Natural source code statement and the corresponding object code in the cataloged object. Therefore, multiple Natural statements can be merged into one NS event and conversely, one Natural statement can cover multiple NS events. |
| **I**nbound **R**PC Message | RI | R | When the Natural RPC server layer receives the client request. |
| **S**tart of **R**PC Request Execution | RS | R | When the Natural RPC server layer calls the Natural server program. |
| **O**utbound **R**PC Message | RO | R | When the Natural RPC server returns the result to the client. |
| **R**PC **W**ait for Client | RW | R | When the Natural RPC server waits for the next message from the client. |
| **U**ser-Defined **E**vent | U | U | When a user-defined event was generated. See ***Writing User-Defined Events***. |

| Event | Event Type | Event Group | When the Event Occurs |
|---|---|---|---|
| **M**onitor **P**ause | MP | M | When the data collection is paused.<br><br>A pause event can be caused by an explicit pause request, at the start of a block filter or when the data pool is full. The duration of a pause is not considered for the application performance analysis. |

With each collected event, a CPU and an event timestamp are recorded. In general, a timestamp is taken at the beginning of an event. The duration of an event therefore equals the time that elapses between the timestamp of the event and the timestamp of the event that follows.

## Data Collected

This section describes the data collected by the Profiler utility:

**General Data**

The following data elements are collected at every event:

- Event counter
- Event type
- Event time in units of microseconds
- Session CPU time in units of microseconds
- Trace session ID
- Natural Security user group ID
- Natural user ID
- Natural application name
- Program library
- Program name
- Program level
- Copycode library
- Copycode name
- Statement line number
- Statement op-code
- Coverage flag (for Natural code coverage)

**Notes:**

1. The time spent for data collection (Monitor CPU time) is measured separately and extracted from the session CPU time.
2. The events are counted before any filtering or sampling is performed. Therefore, the events receive in general the same counting results regardless of which filtering or sampling is used.
3. Statement events are only counted if the event filter keyword `STATEMENT=COUNT` is used.
4. Monitor Pause events are not counted.
5. Natural code coverage only collects `NS` and `PI` events.
6. Natural code coverage does not collect time values.
7. A `PI` event is collected for each object accessed and for all copycodes included in the object (recursively).

**Event-Specific Data**

The following data is only collected at the following events:

| Event | Data Elements |
|---|---|
| Session Initialization | None |
| Session Termination | Termination return code<br>Natural termination message code NAT99*nn*<br>Name of back-end program<br>Monitor CPU time in units of microseconds |
| Program Load | Name of program to be loaded<br>Name of load library<br>Invocation type |
| Program Resume | None |
| Program Start/Termination | Program type<br>Database ID of program library<br>File number of program library |
| Program Information | Program type<br>Number of statements in the program or copycode<br>First statement item<br>`INCLUDE` line number<br>Parent copycode ID |
| Database Call | Database type<br>Command code<br>Command ID<br>Database ID<br>File number<br>Response code (event type `DA`)<br>Error subcode (event type `DA`)<br>Adabas command time (event type `DA`) |
| Terminal I/O | Number of bytes sent<br>Number of bytes read |

| Event | Data Elements | |
|---|---|---|
| | Total session storage allocated<br>Compressed session storage length | |
| External Program Call | Name of program called<br>Calling mode such as dynamic or static mode<br>Program link location<br>Parameter type such as reference or value<br>Response code (event type CA) | |
| Runtime Error | Natural system error message code<br>External abend code<br>Name of error handling program | |
| Natural Statement | Profiling: None<br><br>Natural code coverage: Statement item identifier (GP offset) | |
| Start of RPC Request Execution | Environment (C = client, S = server)<br>Subprogram name<br>Adabas user ID (ETID)<br>Conversation status<br>Logon indicator (Y = logon performed)<br>Impersonation indicator of RPC request (Y = impersonation performed) | |
| Outbound/Inbound RPC Message / RPC Wait for Client | Environment (C = client, S = server)<br>Transport protocol<br>RPC function<br>Type of client user ID<br>Length of message<br>RPC return code<br>External conversation ID<br>Client user ID<br>Server node (event types RO and RW)<br>Server name (event types RO and RW) | |
| User-Defined Event | Subtype of the user-defined event<br>Up to 249 bytes of user-defined information | |
| Monitor Pause | Type of monitor pause<br><br>Possible values: | |
| | R | Monitor pause requested. This value is also set when the session is initialized with the Pause option. |
| | F | Start of a block of filtered-out events. Block filters are: library, program, line, FNAT, event count, or time filter. |
| | W | Trace session waits because of a data pool full situation. |

## Initializing Profiling

The Profiler utility `INIT` function initializes profiling. The `INIT` function must be performed before any of the Profiler utility functions `START`, `PAUSE` or `TEXT` or any Profiler filter.

Syntax of `INIT`:

```
FUNCTION=INIT
 [RESOURCE={ON|OFF}]
 [RESOURCE-NAME=resource-name]
 [RESOURCE-LIB=library-name]
 [REPLACE={YES|NO}]
 [SAMPLING={ON|OFF}]
 [INTERVAL={100|interval-time}]
 [CONSOLIDATE={ON|OFF}]
 [WAIT-FULL={60|wait-full-time}]
 [WAIT-EMPTY={60|wait-empty-time}]
 [CMPRMIN=data-set-name]
 [TRACE-EVENT={ON|OFF}]
 [TRACE-MONITOR={3|trace-level}]
 [TRACE-CONSOLIDATE={ON|OFF}]
```

Syntax Description:

| Keyword for INIT | Value | Description |
|---|---|---|
| `RESOURCE` | | Specifies whether the event data is to be written to a Natural Profiler resource file (NPRF or NPRC). |
| | `ON` | The event data is written to a Natural Profiler resource file and can be analyzed with NaturalONE or with the data processing functions of the Natural Profiler utility. |
| | `OFF` | The event data is not written to a Natural Profiler resource file. This option can be used if only the Profiler event trace or the statistics are required. |
| `RESOURCE-NAME` | *`resource-name`* | The name of the Profiler resource file in which the event data is saved for a later analysis. If the extension `.nprf` (Natural Profiler resource file) or `.nprc` (Natural Profiler resource consolidated) has not been specified, it is added automatically. Which extension is used depends on the `CONSOLIDATE` keyword.<br><br>Default: `*INIT-USER_`*`yyyymmdd_hhiiss` |

| Keyword for INIT | Value | Description |
|---|---|---|
| | | where `*INIT-USER` is the content of the corresponding Natural system variable (name of the job or user ID under which the Natural session is running);<br><br>and *`yyyymmdd_hhiiss`* is the resource allocation date and time. |
| `RESOURCE-LIB` | *`library-name`* | The name of the Natural library in which the resource is allocated.<br><br>Default: The name of the current library |
| `REPLACE` | | Specifies whether the resource is replaced if it already exists. |
| | `YES` | Replace the resource if it exists. |
| | `NO` | Do not replace the resource. If the resource already exists, a message is written and no profiling is performed. |
| `SAMPLING` | | Specifies whether the Profiler CPU-time sampling is activated. Sampling can reduce dramatically the number of monitored events whereby it achieves nearly the same CPU-time results. See ***Sampling***. |
| | `ON` | Activate sampling. |
| | `OFF` | Deactivate sampling. |
| `INTERVAL` | *`interval-time`* | The sampling interval determines at which CPU timestamp events are collected. If the sampling interval is greater, fewer events are collected. If the sampling interval is smaller, the resulting CPU times are more accurate.<br><br>Valid values for *`interval-time`*:<br><br>`1` to `2147483647`<br><br>Unit: microseconds |
| `CONSOLIDATE` | | Specifies whether the event data is consolidated (aggregated) before it is written to the resource file. See ***Data Consolidation and Processing*** in the section *Basic Concepts of the Profiler Utility*. |
| | `ON` | The event data is consolidated and written to an NPRC (Natural Profiler resource consolidated) resource file. |
| | `OFF` | The event data is written unconsolidated to an NPRF (Natural Profiler resource file) resource file. |
| `WAIT-FULL` | *`wait-full-time`* | Specifies how long the trace session waits if the Profiler data pool is full. If the limit is reached and there is still no space released, Natural terminates (`ABEND SOC1`).<br><br>The value prevents the trace session from endless waiting on an unexpectedly failed monitor session.<br><br>Valid values for *`wait-full-time`*:<br><br>`1` to `32767` |

| Keyword for INIT | Value | Description |
|---|---|---|
| | | Unit: seconds |
| WAIT-EMPTY | *wait-empty-time* | Specifies how long the Profiler monitor session waits if the Profiler data pool is empty and the trace session is still active. If the limit is reached, a message is written and the profiling is stopped.<br><br>A value of zero (0) means that the Profiler waits without limit. A positive value prevents the monitor session from endless waiting on a potentially faulty trace session.<br><br>Valid values for *wait-empty-time*:<br><br>0 to 2147483647<br><br>When profiling a batch Natural RPC server or profiling a mainframe session from Natural Studio, the value is always set to 0. See ***Profiling a Batch Natural RPC Server*** and ***Profiling a Mainframe Session from Natural Studio***.<br><br>Unit: seconds |
| CMPRMIN | *data-set-name* | Specifies the name of the dynamic parameter input data set for the Profiler monitor session. See ***Monitor Session CMPRMIN***.<br><br>Default: CMPRMIN (the monitor session uses the parameters of the trace session) |
| TRACE-EVENT | | Specifies whether the Profiler event trace is written to the standard output of the Profiler monitor session (MONPRINT data set). See ***Event Trace***. |
| | ON | Write the Profiler event trace. |
| | OFF | Do not write the Profiler event trace. |
| TRACE-MONITOR | *trace-level* | Set the level of the internal trace of the Profiler monitor session. The internal trace contains information such as Profiler errors and is written to the standard output of the monitor session (MONPRINT data set). See ***Internal Trace***.<br><br>Valid trace levels: 0 to 10 |
| TRACE-CONSOLIDATE | | Specifies whether the Profiler consolidation trace is written to standard output. The consolidation trace can only be written if CONSOLIDATE=ON is specified. See ***Consolidation Trace***. |
| | ON | Write the Profiler consolidation trace. |
| | OFF | Do not write the Profiler consolidation trace. |

### Example of INIT

```
FUNCTION=INIT           /* Initialize Profiling
  RESOURCE=ON           /* Write to resource file
  RESOURCE-NAME='Test'  /* Resource name
  RESOURCE-LIB=PRFDATA  /* Resource library
  REPLACE=YES           /* Replace resource
  SAMPLING=ON           /* Use sampling
  INTERVAL=100          /* Sampling interval
  CONSOLIDATE=OFF       /* Do not consolidate the data
  WAIT-FULL=60          /* Wait sec if pool full
  WAIT-EMPTY=60         /* Wait sec if pool empty
  CMPRMIN=PRFPARMS      /* CMPRMIN for monitor session
  TRACE-EVENT=ON        /* Trace events
  TRACE-MON=3           /* Trace level monitor session
```

## Initializing Code Coverage

The Profiler utility `COVERAGE` function initializes the Natural code coverage. The `COVERAGE` function must be performed before any of the Profiler utility functions `START`, `PAUSE` or `TEXT` or any Profiler filter.

Syntax of `COVERAGE`:

```
FUNCTION=COVERAGE
 [RESOURCE={ON|OFF}]
 [RESOURCE-NAME=resource-name]
 [RESOURCE-LIB=library-name]
 [REPLACE={YES|NO}]
 [WAIT-FULL={60|wait-full-time}]
 [WAIT-EMPTY={60|wait-empty-time}]
 [CMPRMIN=data-set-name]
 [TRACE-EVENT={ON|OFF}]
 [TRACE-MONITOR={3|trace-level}]
 [TRACE-COVERAGE={ON|OFF}]
```

Syntax Description:

| Keyword for COVERAGE | Value | Description |
|---|---|---|
| RESOURCE | | Specifies whether code coverage data is to be written to a Natural code coverage resource file (NCVF). |
| | ON | Code coverage data is written to a Natural code coverage resource file and can be analyzed with NaturalONE or with the data processing functions of the Natural Profiler utility. |
| | OFF | Code coverage data is not written to a Natural code coverage resource file. This option can be used if only the Profiler event trace or the statistics are required. |
| RESOURCE-NAME | *resource-name* | The name of the Natural code coverage resource file in which code coverage data is saved for a later analysis. If the extension .ncvf (Natural code coverage resource file) has not been specified, it is added automatically.<br><br>Default: *INIT-USER_*yyyymmdd_hhiiss*<br><br>where *INIT-USER is the content of the corresponding Natural system variable (name of the job or user ID under which the Natural session is running);<br><br>and *yyyymmdd_hhiiss* is the resource allocation date and time. |
| RESOURCE-LIB | *library-name* | The name of the Natural library in which the resource is allocated.<br><br>Default: The name of the current library |
| REPLACE | | Specifies whether the resource is replaced if it already exists. |
| | YES | Replace the resource if it exists. |
| | NO | Do not replace the resource. If the resource already exists, a message is written and no code coverage is performed. |
| WAIT-FULL | *wait-full-time* | Specifies how long the trace session waits if the Profiler data pool is full. If the limit is reached and there is still no space released, Natural terminates (ABEND SOC1).<br><br>The value prevents the trace session from endless waiting on an unexpectedly failed monitor session.<br><br>Valid values for *wait-full-time*:<br><br>1 to 32767<br><br>Unit: seconds |
| WAIT-EMPTY | *wait-empty-time* | Specifies how long the Profiler monitor session waits if the Profiler data pool is empty and the trace session is still active. If the limit is reached, a message is written and code coverage is stopped.<br><br>A value of zero (0) means that the Profiler waits without limit. A positive value prevents the monitor session from endless waiting on a potentially faulty trace session. |

| Keyword for COVERAGE | Value | Description |
|---|---|---|
| | | Valid values for *wait-empty-time*:<br><br>0 to 2147483647<br><br>When covering a batch Natural RPC server or a mainframe session from Natural Studio, the value is always set to 0. See ***Profiling a Batch Natural RPC Server*** and ***Profiling a Mainframe Session from Natural Studio***.<br><br>Unit: seconds |
| CMPRMIN | *data-set-name* | Specifies the name of the dynamic parameter input data set for the Profiler monitor session. See ***Monitor Session CMPRMIN***.<br><br>Default: CMPRMIN (the monitor session uses the parameters of the trace session) |
| TRACE-EVENT | | Specifies whether the Profiler event trace is written to the standard output of the Profiler monitor session (MONPRINT data set). See ***Tracing Natural Code Coverage***. |
| | ON | Write the Profiler event trace. |
| | OFF | Do not write the Profiler event trace. |
| TRACE-MONITOR | *trace-level* | Set the level of the internal trace of the Profiler monitor session. The internal trace contains information such as Profiler errors and is written to the standard output of the monitor session (MONPRINT data set). See ***Internal Trace***.<br><br>Valid trace levels: 0 to 10 |
| TRACE-COVERAGE | | Specifies whether the Profiler coverage trace is written to standard output. See ***Tracing Natural Code Coverage***. |
| | ON | Write the Profiler coverage trace. |
| | OFF | Do not write the Profiler coverage trace. |

## Example of COVERAGE

```
FUNCTION=COVERAGE        /* Initialize code coverage
  RESOURCE=ON            /* Write to resource file
  RESOURCE-NAME='Test'   /* Resource name
  RESOURCE-LIB=COVDATA   /* Resource library
  REPLACE=YES            /* Replace resource
  WAIT-FULL=60           /* Wait sec if pool full
  WAIT-EMPTY=60          /* Wait sec if pool empty
  CMPRMIN=PRFPARMS       /* CMPRMIN for monitor session
  TRACE-EVENT=ON         /* Trace events
  TRACE-MON=3            /* Trace level of monitor session
  TRACE-COVERAGE         /* Trace coverage data
```

## Starting and Pausing Data Collection

The Profiler must be initialized before the data collection can be started or paused. Because the data collection is paused after the initialization, it has to be started in any way so that event data is recorded.

You can start and pause data collection with the following methods:

- Using Profiler Utility Functions
- Using Profiler Utility Programs
- Using the Application Programming Interface

### Using Profiler Utility Functions

The Profiler utility START and PAUSE functions are used to start and pause data collection. The following syntax applies:

```
FUNCTION=START [COUNT={0|count-number}]
FUNCTION=PAUSE
```

Syntax Description:

| Keyword for START | Value | Description |
|---|---|---|
| COUNT | *count-number* | Set the event counter of the next monitored event to the specified value.<br><br>Valid values for *count-number*:<br><br>0 to 2147483647<br><br>The event counter remains unchanged if a value of zero (0) is specified. |

### Using Profiler Utility Programs

The following Natural programs in the system library SYSPRFLR are supplied to perform Profiler utility functions:

| Program | Description |
|---|---|
| PRFSTART | Start the data collection. |
| PRFPAUSE | Pause the data collection. |
| PRFSTATE | Get the state of the data collection. |
| PRFFCT | Execute a Profiler utility function: START, PAUSE or STATE. |

### To use Profiler utility programs

- Logon to the library SYSPRFLR or copy the programs to the library SYSTEM, to the appropriate steplib library, or to the required library.

  If PRFFCT is used, the application programming interface USR8210N has to be copied as well (see the following section).

  If PRFFCT is used in a client/server environment, copy PRFFCT to the client library and USR8210N to the server library.

  **Note:** PRFFCT expects as input the value START, PAUSE or STATE to perform the corresponding function.

### To start the data collection

- Execute the following program:

```
PRFSTART
```

  Or:

```
PRFFCT
START
```

### To pause the data collection

- Execute the following program:

```
PRFPAUSE
```

  Or:

```
PRFFCT
PAUSE
```

### To retrieve the current state of the data collection

- Execute the following program:

```
PRFSTATE
```

Or:

```
PRFFCT
STATE
```

## Using the Application Programming Interface

The data collection can be started and paused from the profiled Natural application by calling the application programming interface (API) `USR8210N`. The API can also be used to get the current state of the monitoring process. The API is delivered in the `SYSEXT` library. For more information, see *SYSEXT Utility - Natural Application Programming Interfaces*.

### To use the API

- Copy the subprogram `USR8210N` to the library `SYSTEM`, to the appropriate steplib library, or to the required library.

  **Note:** `USR8210N` expects as the first parameter the value `START`, `PAUSE` or `STATE` to perform the corresponding function. The parameter values can be specified in uppercase or lowercase. On return, `P-RETURN` contains the return code and `P-MESSAGE` the success or error message.

### To start the data collection

- Use the interface with the `CALLNAT` statement:

```
CALLNAT 'USR8210N' 'START' P-RETURN P-MESSAGE /* Start Profiler
```

### To pause the data collection

- Use the interface with the `CALLNAT` statement:

```
CALLNAT 'USR8210N' 'PAUSE' P-RETURN P-MESSAGE /* Pause Profiler
```

### To retrieve the current state of the data collection

- Use the interface with the `CALLNAT` statement:

```
CALLNAT 'USR8210N' 'STATE' P-RETURN P-MESSAGE /* Get Profiler state
```

The state is coded in the field `P-RETURN`:

| P-RETURN | Description |
|---|---|
| 0 | Natural Profiler data collection is started. |
| 1 | Natural Profiler data collection is paused. |

## Using Filters to Limit the Data Collected

Filters play an important role in reducing the amount of collected data. The following filters are available:

- Event Filter
- Program Filter
- Count Filter
- Time Filter

**Note:** The Profiler must be initialized before any settings can be applied to a specific filter.

### Event Filter

The Profiler event filter specifies which events are collected. Additionally, it determines whether event data is recorded while Natural system programs are executing.

Syntax of Event Filter:

```
FILTER=EVENT
 [EVENT={[event-type]...|ALL}
 [STATEMENT={ON|OFF|COUNT}]
 [FNAT={ON|OFF}
```

Syntax Description:

| Keyword for EVENT | Value | Description |
|---|---|---|
| EVENT | | Specifies which events are collected. |

| Keyword for EVENT | Value | Description |
|---|---|---|
| | *event-type* | Each event is encoded by a one or two letter code. Multiple events are separated by blanks. Only the specified events are recorded. If no event is given, the Profiler monitors only the session and pause events.<br><br>Possible event entries are: DB, DA, PL, PS, PT, PR, IB, IA, E, CB, CA, U, RS, RI, RO and RW.<br><br>**Notes:**<br><br>1. For information on the event codes, see ***Events and Data Collected***.<br>2. The following events cannot be specified in the event list:<br>■ SI (session initialization) events cannot be collected by the Profiler in batch mode,<br>■ ST (session termination) events and MP (Monitor Pause) events are always collected,<br>■ The collection of NS (Natural Statement) events depends on the STATEMENT keyword.<br>3. If only one character is specified, then all events beginning with this character are recorded. For example, EVENT=P is equivalent to EVENT=PL PR PS PT. |
| | ALL | All events are recorded; that is, the definition is equivalent to<br><br>EVENT=DB DA PL PS PT PR IB IA E CB CA U RS RI RO<br><br>**Caution:** EVENT=ALL does not include the NS event. The collection of NS events (Natural Statement) depends on the STATEMENT keyword. |
| STATEMENT | | Specifies whether Natural statement (NS) events are collected. |
| | ON | Natural statement (NS) events are collected but not counted. Only non-statement events are counted. Natural statement events receive the same count as the preceding event. |
| | OFF | Natural statement (NS) events are not collected. |
| | COUNT | Natural statement (NS) events are collected and counted, that means, the event counter field in the event record is incremented with each statement event. This option can lead to poorer performance. See ***Profiler Performance in Batch***. |
| FNAT | | Specifies whether event data is recorded while Natural system programs are executing. |
| | ON | Event data is recorded while Natural system programs are executing. |
| | OFF | Event data is not recorded while Natural system programs are executing. |

### Default Filter Value for Profiling

By default (if the event filter is not specified), all events except Natural statement events are collected and event data of Natural system programs is not recorded:

```
FILTER=EVENT            /* Set event filter
  EVENT=ALL             /* All events
  STATEMENT=OFF         /* Do not collect statements
  FNAT=OFF              /* No FNAT
```

**Default Filter Value for Code Coverage**

By default (if the event filter is not specified), only program start and Natural statement events are collected. Event data of Natural system programs is not recorded:

```
FILTER=EVENT            /* Set event filter
  EVENT=PS              /* Program start only
  STATEMENT=ON          /* Collect statements
  FNAT=OFF              /* No FNAT
```

Code coverage will not work correctly if program start or Natural statement events are not collected. If other events are monitored, they will be displayed in the trace but ignored by code coverage.

**Example of an Event Filter**

```
FILTER=EVENT            /* Set event filter
  EVENT=D PS PR         /* Database and program start/resume events
  STATEMENT=ON          /* Collect statements (no count)
  FNAT=OFF              /* No FNAT
```

### Program Filter

The Profiler program filter specifies the libraries, programs (Natural objects) and program lines for which event data is collected. By default (if the program filter is not specified), the data of all libraries, programs and lines is collected.

Syntax of Program Filter:

```
FILTER=PROGRAM
 [LIBRARY=library-name]
 [PROGRAM=program-name]
 [LINE-FROM={0|start-number}
 [LINE-TO={0|end-number}
```

Syntax Description:

| Keyword for PROGRAM | Value | Description |
|---|---|---|
| LIBRARY | *library-name* | Only the specified library is monitored. If the specification ends with an asterisk (*), all libraries with the corresponding prefix are monitored.<br><br>Default: all libraries |
| PROGRAM | *program-name* | Only the specified program is monitored. If the specification ends with an asterisk (*), all programs with the corresponding prefix are monitored.<br><br>Default: all programs |
| LINE-FROM | *start-number* | Only lines with a line number greater or equal the specified number are monitored.<br><br>Valid values for *start-number*:<br><br>0 to 9999 |
| LINE-TO | *end-number* | Only lines with a line number less or equal the specified number are monitored. If the number is 0, the maximum line number is used.<br><br>Valid values for *end-number*:<br><br>0 to 9999 |

### Example of a Program Filter

The following example monitors the lines 0500 to 2000 in all Natural objects starting with X on the library PRFDEMO.

```
FILTER=PROGRAM            /* Set program filter
  LIBRARY=PRFDEMO         /* Monitored library
  PROGRAM=X*              /* Monitored program
  LINE-FROM=500           /* Monitor from line 0500
  LINE-TO=2000            /* Monitor to line 2000
```

## Count Filter

The Profiler count filter specifies the event counters for which data is collected. By default (if the count filter is not specified) the data of any event count is collected.

If the event filter STATEMENT=ON is set, the count filter can only refer to non-statement events because statement events do not get a unique count. The non-statement events have the same count as with STATEMENT=OFF and the same count filter can be used in both cases.

Syntax of Count Filter:

```
FILTER=COUNT
 [FROM={0|minimum-count}
 [TO={0|maximum-count}
```

Syntax Description:

| Keyword for COUNT | Value | Description |
|---|---|---|
| FROM | *minimum-count* | Only events with an event counter greater or equal the specified number are monitored.<br><br>Valid values for *minimum-count*:<br><br>0 to 2147483647 |
| TO | *maximum-count* | Only events with an event counter less or equal the specified number are monitored. If the number is 0, the maximum event counter is used.<br><br>Valid values for *maximum-count*:<br><br>0 to 2147483647 |

**Example of a Count Filter**

A profiling with STATEMENT=OFF has shown that a lot of CPU time was spent between the events with the counters 1200 to 1400. Now, we want to analyze this range in more detail including the statements. With STATEMENT=COUNT, the statements would be counted as well and the events would receive other counter values. But with STATEMENT=ON the statements are not counted and the counter values can be used to restrict the data collection.

```
FILTER=COUNT            /* Set count filter
  FROM=1200             /* Monitor from event count 1200
  TO=1400               /* Monitor to event count 1400
```

### Time Filter

The Profiler time filter specifies the CPU-times (in units of 1/100 sec) for which data is collected. By default (if the time filter is not specified), the data of any CPU time is collected.

Syntax of Time Filter:

```
FILTER=TIME
 [FROM={0|minimum-time}
 [TO={0|maximum-time}
```

Syntax Description:

| Keyword for TIME | Value | Description |
|---|---|---|
| FROM | *minimum-time* | Only lines with a CPU-time greater or equal the given number of sec/100 are monitored.<br><br>Valid values for *minimum-time*:<br><br>0 to 2147483647 |
| TO | *maximum-time* | Only lines with a CPU-time less or equal the given number of sec/100 are monitored. If the number is 0, the maximum CPU-time is used.<br><br>Valid values for *maximum-time*:<br><br>0 to 2147483647 |

**Example of a Time Filter**

The following example monitors all events which occur after one second for the duration of two seconds.

```
FILTER=TIME             /* Set time filter
  FROM=100              /* Monitor from CPU second 1.00
  TO=300                /* Monitor to CPU second 3.00
```

## Enabling Sampling

The sampling method uses a statistical approach to collect data. Sampling significantly reduces the amount of data written to the resource file while approximately retaining the same CPU times as without sampling.

Sampling is available for profiling but not for code coverage.

For general information regarding sampling, see ***Sampling*** in the section *Basic Concepts of the Profiler Utility*.

### To enable sampling

- Enter the following subordinate keywords associated to the Profiler utility INIT function:

```
SAMPLING=ON
INTERVAL=nnn
```

where *nnn* is the sampling interval in microseconds.

**Note:** By default (if SAMPLING is not specified), the data is not sampled. If SAMPLING=ON is specified but no INTERVAL, the default sampling interval is 100 microseconds.

## Writing User-Defined Events

A user-defined event can be generated from a Natural program by using the Natural statement

```
CALL 'CMRDC' 'U' USER-DATA
```

The first character of the USER-DATA is treated as the subtype of the user-defined event. The remaining characters are the text of the user-defined event. For details, see *User-defined Events* in the *SYSRDC Utility* documentation.

**Notes:**

1. The Natural Profiler in batch mode offers a function to generate user-defined events from the JCL input stream. These user-defined events are always recorded regardless of the current filter settings. If profiling is paused, the Profiler utility activates the profiling before it writes the user-defined event and deactivates it afterwards. If sampling is active, it can happen that a user-defined event is sampled away.
2. The Profiler must be initialized before a user-defined event can be written.

The Profiler utility FUNCTION=TEXT function is used to write user-defined events. The following syntax applies:

```
FUNCTION=TEXT
 [TEXT=text]
 [TYPE=character]
```

Syntax Description:

| Keyword for TEXT | Value | Description |
|---|---|---|
| TEXT | *text* | The *text* is added to the Profiler trace as a user-defined event. If multiple TEXT keywords are specified, the corresponding values are concatenated. The maximum text size is 249 bytes. Additional characters will be truncated.<br><br>Default: none (blanks) |
| TYPE | *character* | The alphanumeric *character* specifies the subtype of the user-defined event. The subtype is part of the event-specific data. The event type of a user-defined event is always U followed by a blank.<br><br>Default: blank |

Use the following event filter setting if you only want to monitor the user-defined events written with the TEXT function. In addition to the TEXT function entries, the session and pause events are also monitored. All other events, including user-defined events written by calls to CMRDC are filtered out.

```
* Monitor only TEXT function entries
FILTER=EVENT/* Set event filter
  EVENT=                /* No events
  STATEMENT=OFF         /* No statements
  FNAT=OFF              /* No FNAT
```

**Example of a User-Defined Event**

The following example writes a user-defined event with subtype J and the text Start profiling into the Profiler event trace.

```
FUNCTION=TEXT            /* Write a user-defined event
  TEXT='Start profiling' /* User-defined event text
  TYPE='J'               /* User-defined event subtype
```

## Monitor Session CMPRMIN

By default, the Profiler monitor session uses the same dynamic Natural parameters as the trace session specified with the CMPRMIN input data set. Exception: the RDCSIZE profile parameter is set to zero (0) for the monitor session. With the CMPRMIN keyword of the Profiler utility INIT or COVERAGE function a separate dynamic parameter input data set can be defined for the Profiler monitor session. If you use the separate dynamic parameter input data set, consider the following:

- Specify for the monitor session only those Natural parameters which are required. Do not specify parameters required for the application execution (such as RPC).
- Specify the RDCSIZE and RDCEXIT parameters only for the trace session. Specifying these parameters for the monitor session will lead to unnecessary calls to the exit and a poorer performance.

- Specify the `PDPSIZE` parameter only for the monitor session. Any specification in the trace session dynamic parameter input data set is ignored.

If the default Natural profile parameter `ETID` setting is used, it can happen that the following error is received by the Profiler monitor session:

```
NAT3048 Error during Open processing. DB/Subcode nn/8 - ETID=job-name.
```

In this case, use the following parameters for `CMPRMIN`:

```
ETID=' ',DBCLOSE=ON
```

Alternatively, you can use an `ETID` value different from the job name in the separate dynamic parameter input data set for the Profiler monitor session.

**To define a dynamic parameter input data set for the Profiler monitor session**

- Enter the following subordinate keyword associated to the Profiler utility `INIT` or `COVERAGE` function:

  ```
  CMPRMIN=data-set
  ```

  where *`data-set`* is the name of the dynamic parameter input data set for the Profiler monitor session.

**Example for z/OS**

```
//CMSYNIN  DD  *
PROFILER
FUNCTION=INIT             /* Initialize profiling
  CMPRMIN=PRFPARMS        /* Monitor session parameter
  ...
/*
//* Trace Session Parameters
//CMPRMIN  DD   *
RDCSIZE=2,RDCEXIT=NATRDC1,...
/*
//* Monitor Session Parameters
//PRFPARMS DD   *
ETID=PROFILER,PDPSIZE=10000,...
/*
```

## Profiling a Batch Natural RPC Server

Profiling or performing code coverage of a batch Natural RPC server requires that the `PROFILER` system command and the Profiler input data are entered with the Natural `STACK` profile parameter in the RPC server job. The Profiler input must be entered in the second syntax format (without equal signs and commas). See also ***Profiler Utility Syntax*** in ***Syntax and Keywords***.

### To start profiling of a batch Natural RPC server

- Enter the following keyword before the Profiler `INIT` or `COVERAGE` function is performed:

```
RPC
```

The RPC keyword indicates the Profiler that a Natural RPC server is monitored. The Profiler overwrites the `WAIT-EMPTY` keyword of the initialization with a value of zero (`0`) so that profiling always continues when the Profiler data pool is empty and the trace session is still active.

We recommend that you use a separate dynamic parameter input data set (***Monitor Session CMPRMIN***) for the Profiler monitor session when profiling a batch Natural RPC server. Do not specify the `RPC` parameter in this data set. See ***Monitor Session CMPRMIN***.

The event data of the batch Natural RPC server is written to a Profiler resource file. The name and library of the resource file can be specified with Profiler keywords (see *Initializing Profiling*). We recommend that you stop the Natural RPC server before the Profiler further processes the resource file.

If you profile a Natural RPC server, you can start and pause data collection with the programs supplied for the Profiler utility.

### To start and pause Profiler data collection in a client/server environment

1. Copy the program `PRFFCT` from the system library `SYSPRFLR` to the client library, and the application programming interface `USR8210N` from the system library `SYSEXT` to the server library.
2. Execute the program `PRFFCT` in the client library.

   `PRFFCT` expects as input the value `START` or `PAUSE` to perform the corresponding function. If you enter the value `STATE`, the current state of the data collection is displayed.

   For more information, see ***Starting and Pausing Data Collection***.

### Example of Natural RPC Batch Profiling

The following example for z/OS shows the Profiler input data for a batch Natural RPC server specified with the Natural STACK profile parameter:

```
STACK=(
PROFILER RPC:
TRACE:3:
FUNCTION:INIT:
  TRACE-EVENT:OFF:
  TRACE-MON:3:
  CMPRMIN:CMPRMINX:
  RESOURCE:ON:
  RESOURCE-NAME:RPCTEST:
  RESOURCE-LIB:PRFDATA:
  REPLACE:YES:
FILTER:EVENT:
  EVENT:ALL:
  STATEMENT:ON:
  FNAT:OFF:
END-PROFILER;
LOGON PRFDEMO
)
```

## Profiling a Mainframe Session from Natural Studio

You can use the Profiler utility in batch mode to profile a mainframe application or run code coverage for a mainframe application that executes remotely from Natural Studio on a Natural Development Server.

This requires that you specify the PROFILER system command and the PROFILER input data with the Natural STACK profile parameter dynamically set when you map your remote mainframe environment. The Profiler input must be entered in the second syntax format (without equal signs and commas). See also ***Profiler Utility Syntax*** in ***Syntax and Keywords***.

You can use the Profiler INCLUDE keyword to read the Profiler input from a Natural text object, and thus reduce the amount of data entered with the STACK parameter. For an example, see the XNDV text object delivered in the SYSPRFLR system library. This example initializes the profiling and immediately starts the monitoring.

The Profiler automatically overwrites the WAIT-EMPTY keyword of the initialization with a value of zero (0) so that profiling always continues when the Profiler data pool is empty and the trace session is still active.

The event data of the Natural Studio mainframe session is written to a Profiler resource file. The name and library of the resource file can be specified with Profiler keywords (see ***Initializing***

*Profiling* or *Initializing Code Coverage*). We recommend that you disconnect the mainframe session before you evaluate the resource file.

If you profile a Natural Studio mainframe session, you can **start and pause data collection** (see the relevant section) with the programs supplied for the Profiler utility.

**To start and pause Profiler data collection in a Natural Studio mainframe session**

1 Logon to the library `SYSPRFLR`.
2 Execute the `PRFSTART` program to start the collection.
3 Execute the `PRFPAUSE` program to pause the collection.

### Rules and Restrictions

The following rules and restrictions apply when profiling a mainframe session from Natural Studio:

- Profiling with the Profiler utility in batch mode does not work if programs are executed remotely from NaturalONE. Use the NaturalONE Profiler if you want to profile mainframe programs executed in a NaturalONE environment.
- Profiling of a Natural Studio mainframe session with the Profiler utility in batch mode does not work if the Natural Development Server uses a CICS adapter.
- If profiling is started with the `STACK` parameter, use the `TRACE=O` Profiler setting.
- The monitor trace (`TRACE-MONITOR`) of the Profiler session is written to the Natural Development Server output.

### Example of STACK for Mainframe Session Profiling

The following is an example of a `STACK` parameter specification for Profiler input data set dynamically when mapping to a remote mainframe environment from Natural Studio:

```
STACK=(PROFILER RESOURCE-LIB:PRFDATA:INCLUDE:XNDV)
```

The resource is written to the `PRFDATA` library.

## Consolidating Event Data

The Profiler utility `CONSOLIDATE` function consolidates event data.

For general information regarding data consolidation, see ***Data Consolidation*** in the section *Basic Concepts of the Profiler Utility*.

Syntax of `CONSOLIDATE`:

```
FUNCTION=CONSOLIDATE
 [RESOURCE={ON|OFF}]
 [RESOURCE-NAME=resource-name]
 [RESOURCE-LIB=library-name]
 [REPLACE={YES|NO}]
 [IO-TIME={ON|OFF}]
 [EXPORT={ON|OFF}]
 [FORMAT={TEXT|COMMA|SEMICOLON}]
 [TRACE-EVENT={ON|OFF}]
 [TRACE-CONSOLIDATE={ON|OFF}]
```

Syntax Description:

| Keyword for CONSOLIDATE | Value | Description |
|---|---|---|
| `RESOURCE` | | Specifies whether the consolidated event data is written to a Natural Profiler resource consolidated (NPRC) resource file. |
| | `ON` | The consolidated event data is written to an NPRC resource file. |
| | `OFF` | The consolidated event data is not written to an NPRC resource file.<br><br>This setting is useful if you want to print the event trace or statistics or export the data and you do not need the consolidated NPRC resource file. |
| `RESOURCE-NAME` | *`resource-name`* | The name of the Natural Profiler resource file (NPRF) you want to consolidate.<br><br>The file extension `.nprf` is added automatically.<br><br>Default: The name of the last created NPRF resource file in the library |

| Keyword for CONSOLIDATE | Value | Description |
|---|---|---|
| | | If RESOURCE=ON, the consolidated data is written to an NPRC resource file with the same resource name. |
| RESOURCE-LIB | *library-name* | The name of the Natural library that contains the NPRF resource file you want to consolidate.<br><br>Default: The name of the current library.<br><br>This library is also used as the target library for the consolidated NPRC resource file. |
| REPLACE | | Specifies whether an existing NPRC resource file is replaced. |
| | YES | Replace an existing NPRC resource file with the same name. |
| | NO | Do not replace an existing NPRC resource file with the same name.<br><br>A message is returned if a resource file with the same name already exists. No consolidation is performed in this case. |
| IO-TIME | | Specifies whether I/O times (IB event) and Natural RPC client times (RW event) are included in the consolidated data. |
| | ON | I/O and Natural RPC client time are included in the consolidated data. |
| | OFF | I/O and Natural RPC client time are not included in the consolidated data.<br><br>This setting is useful if the event data originates from the Profiler utility in online mode and you plan to analyze the performance in NaturalONE or with the Profiler utility READ function. For a performance analysis with the Profiler MashApp this setting is not required because the MashApp offers a similar function. |
| EXPORT | | Specifies whether the consolidated event data is written to Work File 7. |
| | ON | Write to Work File 7. |
| | OFF | Do not write to Work File 7. |
| FORMAT | | Specifies the format in which the exported data is written to Work File 7. |
| | TEXT | Write the data in free text format. |
| | COMMA | Write the data in CSV format with a comma (,) separator. |
| | SEMICOLON | Write the data in CSV format with a semicolon (;) separator. |
| TRACE-EVENT | | Specifies whether the Profiler event trace is written to standard output.<br><br>See ***Event Trace***. |
| | ON | Write the Profiler event trace. |
| | OFF | Do not write the Profiler event trace. |

| Keyword for CONSOLIDATE | Value | Description |
|---|---|---|
| TRACE-CONSOLIDATE | | Specifies whether the Profiler consolidation trace is written to standard output. See *Consolidation Trace*. |
| | ON | Write the Profiler consolidation trace. |
| | OFF | Do not write the Profiler consolidation trace. |

### Example of a Consolidation

The following example consolidates the Profiler resource Test.nprf in the library PRFDATA and writes the consolidated data to the Profiler resource Test.nprc. I/O and Natural RPC client times are included in the consolidated data.

In addition, the consolidated data is written in CSV (semicolon-separated values) format to Work File 7.

The event and consolidation traces are switched off.

```
FUNCTION=CONSOLIDATE    /* Consolidate Profiler data
  RESOURCE=ON           /* Write to resource
  RESOURCE-NAME='Test'  /* Resource name
  RESOURCE-LIB=PRFDATA  /* Resource library
  REPLACE=YES           /* Replace resource
  IO-TIME=ON            /* Include I/O and RPC client times
  EXPORT=ON             /* Write to Work File 7
  FORMAT=S              /* CSV format with semicolon separator
  TRACE-EVENT=OFF       /* No event trace
  TRACE-CONSOLIDATE=OFF /* No consolidation trace
```

## Evaluating Event Data

When a Natural application is profiled, the Natural Profiler utility writes the event data to an NPRF resource file. Consolidated data is stored in an NPRC resource file and coverage data is stored in an NCVF resource file. The Profiler utility READ function reads and evaluates the Profiler resource data and writes the results to standard output or to a Natural work file. The evaluations performed depend on the type of the resource file read as described in the following table:

| Resource File Type | Evaluation | Description |
|---|---|---|
| NPRF | Event trace | Chronological list of the Profiler event data |
| | Program trace | Program flow of the profiled application |
| | Statistics | Statistics of profiling and the NPRF resource file |
| NPRC | Consolidation trace | List of the consolidated data with hit counts and summarized elapsed time and CPU time |
| | Program summary | Table of executed Natural objects<br><br>The table shows which events occurred during object execution and the CPU time spent executing the object. |
| | Statistics | Statistics of profiling, the consolidation and the NPRC resource file |
| NCVF | Statement coverage | List of statements covered in in the source lines<br><br>The list shows the percentage of statement coverage for each statement line in the source of the accessed programs. |
| | Program coverage | Table of code coverage results of executed Natural objects<br><br>The program coverage table lists all Natural objects which have been executed during the coverage run. For each object, it shows the percentage of coverage, the number of covered and missed statements, and the total number of statements. |
| | Statistics | Statistics for profiling, coverage and the NCVF resource file |

This section covers the following topics:


- Profiler Utility READ Function
- Example of READ
- Event Trace
- Consolidation Trace
- Program Trace
- Program Summary
- Program Coverage
- Statement Coverage
- Using a Microsoft Excel Template to Visualize Coverage Results

- Statistics

## Profiler Utility READ Function

The Profiler utility `READ` function reads and evaluates the resource data.

Syntax of `READ`:

```
FUNCTION=READ
 [RESOURCE-NAME=resource-name]
 [RESOURCE-LIB=library-name]
 [RESOURCE-TYPE={NPRF|NPRC|NCVF}]
 [EVENT={ON|OFF}]
 [PROGRAM={ON|OFF}]
 [STATISTICS={ON|OFF}]
 [PRINT={ON|OFF}]
 [EXPORT={ON|OFF}]
 [FORMAT={TEXT|COMMA|SEMICOLON}]
```

Syntax Description:

| Keyword for READ | Value | Description |
|---|---|---|
| `RESOURCE-NAME` | *`resource-name`* | The name of the NPRF, NPRC or NCVF resource file you want to read.<br><br>If no file extension is specified, the extension specified with the keyword `RESOURCE-TYPE` is added automatically.<br><br>Default: The name of the last created NPRF, NPRC or NCVF resource file in the library depending on the `RESOURCE-TYPE` specification |
| `RESOURCE-LIB` | *`library-name`* | The name of the Natural library that contains the NPRF, NPRC or NCVF resource you want to read.<br><br>Default: The name of the current library |
| `RESOURCE-TYPE` | | Specifies the default resource type (extension) to use if no extension is specified with `RESOURCE-NAME`. |
| | `NPRF` | The default resource type is NPRF with extension `.nprf`. |
| | `NPRC` | The default resource type is NPRC with extension `.nprc`. |
| | `NCVF` | The default resource type is NCVF with extension `.ncvf`. |
| `EVENT` | | Specifies whether the Natural Profiler evaluates events.<br><br>See also ***Event Trace***, ***Consolidation Trace*** and ***Statement Coverage***. |
| | `ON` | NPRF: Write the Natural Profiler event trace. |

| Keyword for READ | Value | Description |
|---|---|---|
| | | NPRC: Write the Natural Profiler consolidation trace.<br>NCVF: Write the statement coverage result. |
| | OFF | Do not evaluate events. |
| PROGRAM | | Specifies whether the Natural Profiler evaluates programs.<br><br>See also *Program Trace*, *Program Summary* and *Program Coverage*. |
| | ON | NPRF: Write the Natural Profiler program trace.<br>NPRC: Write the Natural Profiler program summary.<br>NCVF: Write the program coverage table. |
| | OFF | Do not evaluate programs. |
| STATISTICS | | Specifies whether the Natural Profiler writes statistics.<br><br>See also *Profiler Statistics*.<br><br>**Note:** Statistics data for Natural code coverage is not collected on UNIX and Windows. |
| | ON | Write statistics. |
| | OFF | Do not write statistics. |
| PRINT | | Specifies whether the result is written to standard output. |
| | ON | Write to standard output. |
| | OFF | Do not write to standard output. |
| EXPORT | | Specifies whether the evaluated data is written to the Work File 7. |
| | ON | Write to Work File 7. |
| | OFF | Do not write to Work File 7. |
| FORMAT | | Specifies the format in which the exported data is written to Work File 7. |
| | TEXT | Write the data in free text format. |
| | COMMA | Write the data in CSV format with a comma (,) separator. |
| | SEMICOLON | Write the data in CSV format with a semicolon (;) separator. |

## Example of READ

The following example reads the Natural Profiler resource `Test.nprf` in the library `PRFDATA` and writes the event trace, program trace and the Profiler statistics to standard output and to Work File 7 in text format.

```
FUNCTION=READ            /* Read Profiler Data
  RESOURCE-NAME='Test'   /* Resource name
  RESOURCE-LIB=PRFDATA   /* Resource library
  RESOURCE-TYPE=NPRF     /* Use resource type NPRF
  EVENT=ON               /* Print event trace
  PROGRAM=ON             /* Print program trace
  STATISTICS=ON          /* Print statistics
  PRINT=ON               /* Write to standard output
  EXPORT=ON              /* Write to Work File 7
  FORMAT=TEXT            /* Export in text format
```

## Event Trace

If `EVENT=ON` is specified for an NPRF resource file, the Profiler event trace is generated.

The event trace shows the data of each Natural event which occurred while the application executed. The trace can be referenced if detailed information of an event is required. For example, if a Natural error occurred during application execution, the event trace shows the corresponding error number and message.

If the event trace is written to standard output (`PRINT=ON`) or exported in text format (`EXPORT=ON`, `FORMAT=TEXT`), it is similar to the event trace written by the Profiler monitor session while the application was profiled (see *Event Trace*. If the data is exported in CSV (comma-separated values) format, it contains all data fields provided by the Profiler (see *Data Collected*).

**Example of an Event Trace**

The following example shows an extract of an event trace:

```
Natural Profiler Event Trace
----------------------------
     Count Time              CPU-Time (ms) Ev Lev Library  Program  Line CC-Lib   CC-Name  Statement  Local-Data
         0 10:20:58.219911        63.318   MP 003 SYSPRFD  PRBINIT  8350                   Call       Monitor pause requested
       102 10:20:58.277586        76.106   PL 000                   0000                              Execute SYSEDMD/MENU
       103 10:20:58.277591        76.139   PS 001 SYSEDMD  MENU     0000                   PgmStart   00010/02430 Type: P
       103 10:20:58.277594        76.151   NS 001 SYSEDMD  MENU     0250                   Compute    Assign/Compute/Move
       103 10:20:58.277596        76.155   NS 001 SYSEDMD  MENU     0270                   Fetch      Fetch
       104 10:20:58.277598        76.169   DB 001 SYSEDMD  MENU     0270                   Fetch      00010/02430 S1
...
```

Explanations:

- The **Count** column shows the number of the event. Monitor Pause (`MP`) events and Natural Statement (`NS`) events are not counted and receive the number of the previous event.
- The **Time** and **CPU-Time** columns show the event time and the CPU timestamp of the event execution, respectively.
- The event with the number `104` is a Database Before (`DB`) event caused by an Adabas S1 command issued against the file `00010/02430` which was triggered by a `FETCH` statement in the line `0270` of the Natural object `MENU`.

For further explanations of the trace columns and event types, see the sections *Event Trace* and *Events and Data Collected*.

## Consolidation Trace

If `EVENT=ON` is specified for an NPRC resource file, the Natural Profiler consolidation trace is generated. The consolidation trace is also generated if `CONSOLIDATE=ON` and `TRACE-CONSOLIDATE=ON` are set for the Profiler utility `INIT` function, or if `TRACE-CONSOLIDATE=ON` is set for the Profiler utility `CONSOLIDATE` function.

The consolidation trace shows general event data, summarized values of the elapsed time and CPU time and the hit count of the consolidated record. If two trace entries show the same general event data, they have different event-specific data which is not displayed in the consolidation trace.

The consolidated records are used as the basis for further evaluations like the NaturalONE hot spots or the **Natural Profiler MashApp**. The consolidation trace can be used to validate the consolidated data.

If the consolidation trace is written to standard output (`PRINT=ON`), it is similar to the consolidation trace written by the Profiler data consolidation (see ***Consolidating Event Data***. If the data is exported, it contains all consolidated data fields provided by the Profiler.

### Example of a Consolidation Trace

The following example shows an extract of a consolidation trace:

```
Natural Profiler Consolidation Trace
------------------------------------
    Count Ev User     Lev Library  Program  Line CC-Lib   CC-Name  Statement   Hit-Count  Elapsed(ms)      CPU(ms)
        1 DA PRFO82D  000                    0000                                      1       75.692        0.870
        2 DA PRFO82D  000                    0000                                      1        0.002        0.004
        3 DA PRFO82D  000                    0000                                      1        0.006        0.025
        4 NS PRFO82D  006 SYSLIBS  A82CLS   0010 SYSAOSSU C-COPYRT Reset             43        0.043        0.118
        5 NS PRFO82D  006 SYSTEM   NOMSTCS  4360                    End                1        0.000        0.003
        6 PL PRFO82D  006 SYSTEM   NOMSTCS  0970                    Callnat            1        0.008        0.058
        7 PL PRFO82D  006 SYSTEM   NOMSTCS  1020                    Perform            1        0.004        0.017
...
```

Explanations:

- The **Count** column shows the number of the consolidated record.
- The consolidated record `4` shows that the `RESET` statement in the line `0010` of the copycode `C-COPYRT` (embedded in the Natural object `A82CLS`) executed `43` times spending a total elapsed time of `0.043` milliseconds (`ms`) and a total CPU time of `0.118` ms.
- The application was running on a z/OS machine with zIIP (IBM System z Integrated Information Processor). Under this condition, the CPU time can be higher than the elapsed time.

For further explanations of the trace columns and event types, see the sections ***Event Trace*** and ***Events and Data Collected***.

## Program Trace

If `PROGRAM=ON` is specified for an NPRF resource file, the Profiler program trace is generated. The program trace shows the program flow of the profiled application. In general, the program trace exclusively shows program and session events (see ***Events and Data Collected*** for a list of possible event types).

If the program trace is written to standard output (`PRINT=ON`) or exported in text format (`EXPORT=ON`, `FORMAT=TEXT`), the program names are indented (see the example below) according to the program level to provide a quick overview of the application calling structure.

If the data is exported in CSV (comma-separated values) format, the program names are not intended. In addition to the output in text format, the exported data contains the CPU timestamp and the summarized Adabas time.

**Example of a Program Trace**

The following example shows an extract of a program trace and the totals of the application run:

```
Natural Profiler Program Trace
------------------------------
Time            Ev Library  CC-Name  Line Lev Program  Events
10:20:58.309812 PL                   0000 000
10:20:58.309817 PS SYSEDMD           0000 001 .OPTTEST  D=4 N=2
10:20:58.357694 PL SYSEDMD           5620 001 .OPTTEST
10:20:58.357704 PS SYSEDMD           0000 002 ..CALLMON3 N=3
10:20:58.385263 PL SYSEDMD           0980 002 ..CALLMON3
10:20:58.385274 PS SYSEDMD           0000 003 ...OP3DISC  D=3 N=4
10:20:58.412207 PL SYSEDMD           1670 003 ...OP3DISC
10:20:58.412221 PS SYSEDMD           0000 004 ....OPTINFO  N=57
10:20:58.443203 PL SYSEDMD           5830 004 ....OPTINFO
10:20:58.443210 PS SYSEDMD           0000 005 ....:OPTPARM1 D=3 N=19
10:20:58.449549 PL SYSEDMD           1960 005 ....:OPTPARM1
10:20:58.449555 PS SYSEDMD           0000 006 ....:.OPTPARM2 D=3 N=10
10:20:58.458286 PL SYSEDMD           0560 006 ....:.OPTPARM2
10:20:58.458300 PS SYSEDMD           0000 007 ....:..OPTPARM3 N=16
10:20:58.458390 PL SYSEDMD           1530 007 ....:..OPTPARM3
10:20:58.458408 PS SYSLIBS           0000 008 ....:...NAT41004 D=10 C=6 N=7345
10:20:58.471017 PT SYSLIBS           5235 008 ....:...NAT41004
10:20:58.471017 PR SYSEDMD           1530 007 ....:..OPTPARM3 N=2898
10:20:58.473293 PL SYSEDMD           1530 007 ....:..OPTPARM3
10:20:58.473297 PS SYSLIBS           0000 008 ....:...NAT41004 D=5 C=6 N=1416
10:20:58.475581 PT SYSLIBS           5235 008 ....:...NAT41004
10:20:58.475581 PR SYSEDMD           1530 007 ....:..OPTPARM3 N=466
10:20:58.475957 PT SYSEDMD           2190 007 ....:..OPTPARM3
10:20:58.475957 PR SYSEDMD           0560 006 ....:.OPTPARM2 N=283
10:20:58.476187 PT SYSEDMD           0860 006 ....:.OPTPARM2
10:20:58.476187 PR SYSEDMD           1960 005 ....:OPTPARM1 N=42
10:20:58.476222 PT SYSEDMD           7510 005 ....:OPTPARM1
10:20:58.476222 PR SYSEDMD           5830 004 ....OPTINFO  D=3 N=10
10:20:58.497926 PL SYSEDMD           6080 004 ....OPTINFO
10:20:58.521954 PR SYSEDMD           1670 003 ...OP3DISC  N=241
10:21:11.205102 PR SYSEDMD           0980 002 ..CALLMON3 D=7 N=6070
10:21:41.704996 PR SYSEDMD           5620 001 .OPTTEST  D=8 I=3 N=26
10:21:41.731229 PT SYSEDMD           7370 001 .OPTTEST
10:21:41.731229 PR                   0000 000         D=14 I=1
10:21:42.248348 ST                   0000 000

Totals
------
Ev Event                    Count
S  Session ................ 1
P  Program ................ 5297
D  Database Call .......... 2140
I  Terminal I/O ........... 12
C  External Program Call .. 6510
E  Runtime Error .......... 43
N  Natural Statement ...... 857384
R  RPC Request............. 0
U  User-Defined Event ..... 0
M  Monitor Pause .......... 2
```

Explanations:

- For each event listed, the time when the event occurred, the active library, program (Natural object), copycode, line number and program level is displayed.

- The program name is followed by the number of events that occurred from one program event to the next program event.
- Events which belong to one event group are combined into one count using the maximum count of the corresponding event types. Example: One Database Before (`DB`) and one Database After (`DA`) event are combined into one Database event (`D=1`).
- In the example above, the Natural object `OPTTEST` was started at the level `1`. This program calls the subprogram `CALLMON3` which calls further subprograms. The highest Level 8 is reached when the subprogram `NAT41004` executes. During the first execution, this subprogram performs 10 database calls (`D=10`), 6 external program calls (`C=6`) and 7345 Natural statements (`N=7345`).
- The `Totals` section at the end of the program trace shows the maximum count of each event group. For example: a total of `2140` database calls corresponds to 2140 Database Before (`DB`) and 2140 Database After (`DA`) events.
- The totals of the Session (`S`) and Program (`P`) event groups are only listed under `Totals`; they are not listed next to the program name.

For further explanations of the trace columns, see the section ***Event Trace***.

For explanations of event types and associated event groups, see the section ***Events***.

### Program Summary

If `PROGRAM=ON` is specified for an NPRC resource file, the Profiler program summary is generated.

The program summary shows for each Natural object how many Natural events have occurred, the total CPU time (in units of milliseconds) and the percentage of the CPU time spent by the Natural object with respect to the total CPU time.

Monitor Pause events and events at Level 0 are not taken into account for the program summary. Events which belong to one event group are combined into one count: see ***Events***.

Program starts and load requests are listed separately.

If the data is exported in CSV (comma-separated values) format, the count of each event type is listed. Additionally, the elapsed time and the Adabas times (absolute and percentage values) are displayed. The exported time values are indicated in units of µs (microseconds).

**Example of a Program Summary**

The following example shows an extract of a program summary:

```
Natural Profiler Program Summary
--------------------------------
Library  Program      Start      Load  Database       I/O  External     Error Statement      User CPU-Time (ms)  CPU %
SYSEDMD  ADA-CL          41         0        40         0        41         0       621         0         3.785   0.14
SYSEDMD  ADA-RC          45         0        44         0        45         0       545         0         4.704   0.17
SYSEDMD  AOS-CL         115        97        15         0         0         0      2507         0        42.890   1.63
SYSEDMD  AOS-OP         169       154        22         0         0         0      6975         0        70.286   2.68
SYSEDMD  BYTE             1         0         0         0         0         0        11         0         0.034   0.00
SYSEDMD  CALLMON3         1         5        23         0         0         0      7089         0        20.001   0.76
SYSEDMD  CALLNOM          6         6        19         0         0         0        18         0         1.342   0.05
SYSEDMD  CALLNOPM         2         2         4         0         0         0        16         0         0.395   0.01
SYSEDMD  CALLNOPN         1         1         4         0         0         0         8         0         0.244   0.00
SYSEDMD  CALLNOPS         3         4        23         0         0         1        31         0         1.841   0.07
SYSEDMD  DISNOP           1         7         6         0         0         0       515         0         2.260   0.08
SYSEDMD  DISNO4I          1        47         3         0         1         0      8075         0        25.516   0.97
SYSEDMD  DISNO4IS        57         0         0         0       624         0     36877         0       105.650   4.03
SYSEDMD  DISNRS           1         0         0         0        44         0       511         0         3.343   0.12
SYSEDMD  DISNSP           1        18        15         0         0         0      1850         0         6.074   0.23
SYSEDMD  DISNTMZ          1         4        11         0         0         0       324         0         2.309   0.08
SYSEDMD  MENU             1         1         3         0         0         0         2         0         0.235   0.00
SYSEDMD  MONACSH          1         6         6         0         0         0      1217         0         3.470   0.13
SYSEDMD  MONADA           1      3176        71         0         0         0    272180         0       680.214  25.98
SYSEDMD  MONAREP          1         9        28         0         0         0      1964         0         6.378   0.24
...
Total                  5294      5293      2122         7      6510        43    857384         0      2617.326 100.00
```

Explanations:

- The Natural object `MONADA` consumed the most CPU time: `680.214` ms which corresponds to `25.98` percent of the total CPU time.
- `MONADA` was started once, it loaded `3176` other Natural objects, performed `71` database calls and `272180` Natural statements. There was no I/O, no external call and no error in the program.
- At the end of the program summary, the `Total` counts of the profiling are listed.

### Program Coverage

The program coverage table is generated if `PROGRAM=ON` is specified for an NCVF resource file.

The program coverage table shows the code coverage results for each accessed Natural object. If the table is given in text format, only the GP coverage results (copycodes included) are displayed. In CSV (comma-separated values) format, the table shows lines containing copycode values, additional columns with source counters (copycodes not included) and information regarding `INCLUDE` statements.

In text format, the table provides the coverage count for each accessed library and for the whole application.

The table contains the following columns:

| Column | Description | |
|---|---|---|
| Evaluation | The type of evaluation. Possible types are: | |
| | Program | For program coverage data |
| | Event | For statement coverage data |
| | Statistics | For Profiler statistics data |
| Object Count | The count of cataloged objects (GPs) listed in the table. | |
| Object Type | The type of Natural object such as program and subprogram. | |
| Library | The Natural library that contains the object. | |

| Column | Description |
|---|---|
| Object | The name of the Natural object. |
| Copycode ID | The unique identifier of the copycode instance in the cataloged object (GP). The program gets the copycode ID `0`. |
| Copycode Library | The library from which the copycode is included. |
| Copycode Name | The name of the copycode. |
| GP Coverage% | The percentage of object coverage whereby `INCLUDE` statements are resolved. |
| GP Covered | The number of covered (executed) statements whereby `INCLUDE` statements are resolved. |
| GP Missed | The number of missed (not executed) statements in the object whereby `INCLUDE` statements are resolved. |
| GP Total | The total number of all executable statements in the object whereby `INCLUDE` statements are resolved. |
| Src Coverage% | The percentage of object coverage whereby `INCLUDE` statements are not resolved. |
| Src Covered | The number of covered (executed) statements whereby `INCLUDE` statements are not resolved. |
| Src Missed | The number of missed (not executed) statements in the object whereby `INCLUDE` statements are not resolved. |
| Src Total | The total number of all executable statements in the object whereby `INCLUDE` statements are not resolved. |
| First Statement | The ID of the first statement of the object or copycode. |
| INCLUDE CC-ID | For copycode only.<br><br>The copycode ID of the object or copycode that includes the copycode. |
| INCLUDE Object | For copycode only.<br><br>The name of the object or copycode that includes the copycode. |
| INCLUDE Line | For copycode only.<br><br>The line number of the `INCLUDE` statement that includes the copycode. |

### Example of Program Coverage

The following example shows the result of program coverage in text format:

```
Program Coverage
----------------
Library  Object   Ty Coverage%  Covered    Missed    Total
COVDEMO  TESTCOVN N      84.0%       37         7       44
COVDEMO  TESTCOVP P      69.2%        9         4       13
COVDEMO  -------- --     80.7%       46        11       57
Totals   -------- --     80.7%       46        11       57
```

Explanations:

- The application accesses two objects, the `TESTCOVN` subprogram (`N`) and the `TESTCOVP` program (`P`).
- In `TESTCOVN`, there are 44 executable statements (object code instructions) from which 37 were covered (executed) and 7 missed (not executed), giving a total coverage of 84.0%.
- The summarized values of the two objects accessed in the library `COVDEMO` show coverage of 80.7%.
- Total coverage is also 80.7% because only one library is accessed by the objects.

### Statement Coverage

Statement coverage is generated if `EVENT=ON` is specified for an NCVF resource file.

For statement coverage, the Profiler utility reads the source of the monitored objects. If the source is not found or if the source does not match the collected data, source lines are not printed in the statement coverage report. The Profiler utility resolves `INCLUDE` statements and merges the source of the corresponding copycode into the including program. If the `INCLUDE` structure cannot be resolved, the copycodes are printed separately.

We recommend that you perform the `READ` function soon after the coverage run, as long as the sources correspond to the monitored GPs. As soon as the sources have been modified, the Profiler utility can no longer provide the full information.

Statement coverage shows the percentage of statements covered for each source line of the accessed programs. If the result is written in text format, for each object listed in the statistics, the object coverage values are shown before the statement coverage data. If the result is written in CSV (comma-separated values) format, additional information regarding statement coverage is provided.

The table contains the following columns:

| Column | Description | |
|---|---|---|
| Evaluation | The type of evaluation. Possible types are: | |
| | Program | For program coverage data |
| | Event | For statement coverage data |
| | Statistics | For Profiler statistics data |
| Object Count | The count of objects (GPs) listed in the table. | |
| Library | The Natural library that contains the objects. | |
| Object | The name of the Natural object. | |
| Copycode ID | The unique identifier of the copycode instance in the related cataloged object. The program gets the copycode ID `0`. | |
| Copycode Library | The library that contains the copycode (if copycode is active). | |
| Copycode Name | The name of the copycode (if copycode is active). | |
| Line | The line number in the Natural source object, for example, `0120`. | |

| Column | Description |
|---|---|
| Source | The Natural source line that contains a statement definition, for example, `MOVE #A TO #B`. |
| Coverage% | The percentage of statement coverage of the line. |
| Covered | The number of statements covered (executed) in the line. |
| Missed | The number of missed (not executed) statements in the line. |
| Total | The total number of all executable statements (object code instructions) in the line. |
| Item Coverage | Indicates which statement items (object code instructions) in the line have been covered or missed. Each statement is represented by either `1` or `0`, whereby `1` indicates a covered statement and `0` a missed statement. For example: A value of `x100` indicates that only the first of three statements in the line is covered. |
| Mark | Indicates the coverage state of the line.<br><br>The Mark column can be used to visualize the coverage results in tools like Microsoft Excel. The possible Mark values are listed in ***Using a Microsoft Excel Template to Visualize Coverage Results***. |

### Example of Statement Coverage

The following example assumes that the development has delivered a new version of the `TESTCOVN` subprogram to the quality engineering. After running the test programs, statement coverage of the subprogram shows the following result (text format):

```
Statement Coverage
------------------
M Cov% CC-Lib   CC-Name  Line Source
*                        0010 * Test function Coverage
*                        0020 * Subprogram TESTCOVN
+                        0030 DEFINE DATA
+                        0040 PARAMETER
+                        0050 1 FUNC     (I2)  /* Function
+                        0060 1 RET-CODE (I4)  /* Return code
+                        0070 END-DEFINE
*                        0080 *
*                        0090 /* Return 0 by default
C 100%                   0100 RESET RET-CODE
*                        0110 *
C 100%                   0120 DECIDE ON FIRST VALUE OF FUNC
P  50%                   0130   VALUE 0
M   0%                   0140     PRINT 'Test function 0'
P  66%                   0150   VALUE 1
C 100%                   0160     PRINT 'Test function 1'
C 100%                   0170   VALUE 2
C 100%                   0180     PRINT 'Test function 2'
C 100%                   0190   VALUE 3
C 100%                   0200     PRINT 'Test function 3'
C 100%                   0210   VALUE 4
C 100%                   0220     PRINT 'Test function 4'
```

```
C 100%                    0230   VALUE 5
C 100%                    0240     PRINT 'Test function 5'
C 100%                    0250   VALUE 6
C 100%                    0260     PRINT 'Test function 6'
C 100%                    0270   VALUE 7
C 100%                    0280     PRINT 'Test function 7'
C 100%                    0290   VALUE 8
C 100%                    0300     PRINT 'Test function 8'
P  33%                    0310   VALUE 9
M   0%                    0320     PRINT 'New test function 9'
C 100%                    0330   NONE VALUE
M   0%                    0340     RET-CODE := 1 /* Unsupported function
+                         0350 END-DECIDE
*                         0360 *
C 100%                    0370 END
```

Explanations:

- The Mark (`M`) column shows whether a line is covered (`C`), missed (`M`) or partly covered (`P`).
- No test cases cover the functions `Test function 0` and `New test function 9` (denoted with `M` and 0% coverage). The `NONE VALUE` case is also not covered.
- All other test cases are covered (denoted with `C` and 100% coverage).
- A Natural `VALUE` statement corresponds to multiple object code instructions. The coverage of `50%` and `60%` for `VALUE 0` and `VALUE 1` statements indicates that only a part of these object code instructions are covered.

  This is because one of the generated object code instructions belongs to the previous statement and the others to the current `VALUE` statement for technical reasons.

As a consequence of this coverage analysis, the test cases have to be adjusted so that `Test function 0` and `Test function 9` (and, perhaps, the error case with an unsupported function code) are also covered.

### Using a Microsoft Excel Template to Visualize Coverage Results

Prerequisites: Microsoft Excel and Natural for Windows or Natural for UNIX.

If you want to analyze the coverage result with Microsoft Excel, you can use the Microsoft Excel template delivered with Natural for Windows and Natural for UNIX. Perform the following steps:

1. Perform the Profiler `READ` function and write the output data in CSV (comma-separated values) format to Work File 7. For example:

```
FUNCTION=READ            /* Read Profiler Data
  RESOURCE-NAME='Test'   /* Resource name
  RESOURCE-LIB=PRFDATA   /* Resource library
  RESOURCE-TYPE=NCVF     /* Use resource type NCVF
  EVENT=ON               /* Print statement coverage
  PROGRAM=ON             /* Print program coverage
  STATISTICS=ON          /* Print statistics
  PRINT=ON               /* Write to standard output
  EXPORT=ON              /* Write to Work File 7
  FORMAT=COMMA           /* Export in CSV format
```

2. If your Microsoft Excel requires semicolons as separators, specify the following:

```
FORMAT=SEMICOLON      /* Export in CSV format
```

3. Export the data of Work File 7 with any tool (such as FTP) as a CSV-formatted file to a Windows environment.
4. Open the CSV file with Microsoft Excel.
5. Rearrange the data so that each evaluation type (program, event, statistics) is on its own worksheet in the Microsoft Excel file.
6. Open the delivered template `TESTCOV.XLSX` with Microsoft Excel. The template is contained in the **RES** (Resources) subdirectory of the Natural `SYSPRFLR` system library.
7. For each worksheet, copy the format from the template to your Microsoft Excel:
   - Click on the upper left corner of the table in the template to mark all data in the table.
   - Click on the Microsoft Excel **Copy format** function.
   - Click on the upper left corner of the table in your worksheet to copy the format.

   Now, all entries are formatted as in the template. The source lines are colored and marked as follows:

| Color | Mark | Description |
|---|---|---|
| Green | C | All statements in the line are covered. |
| Yellow | P | The statements in the line are partly covered. |
| Pink | M | All statements in the line are missed. |
| Gray | * | A comment or an empty line. |
| Red | E | Error encountered.<br><br>For example, if the coverage analysis has collected a line number but the corresponding source line is not found. |
| None (white) | + | All other lines such as continuation lines of a statement. |

### Example of a Microsoft Excel Worksheet

The following example shows a worksheet extract of code coverage for the `TESTCOVP` program with included `TESTCOVC` copycode without the columns that contain the object name and library:

| Copycode ID | Copycode Library | Copycode Name | Line | Source | Coverage% | Covered | Missed | Total | Item Coverage | Mark |
|---|---|---|---|---|---|---|---|---|---|---|
| 0 | | | 10 | * Test Coverage | | 0 | 0 | 0 | | * |
| 0 | | | 20 | * Program TESTCOVP | | 0 | 0 | 0 | | * |
| 0 | | | 30 | DEFINE DATA LOCAL | | 0 | 0 | 0 | | + |
| 0 | | | 40 | 1 FUNC (I2) /* function | | 0 | 0 | 0 | | + |
| 0 | | | 50 | 1 RET-CODE (I4) /* Return code | | 0 | 0 | 0 | | + |
| 0 | | | 60 | END-DEFINE | | 0 | 0 | 0 | | + |
| 0 | | | 70 | FOR FUNC = 1 TO 8 | 100 | 4 | 0 | 4 | x1111 | C |
| 0 | | | 80 | /* Test the subprogram functions | | 0 | 0 | 0 | | * |
| 0 | | | 90 | CALLNAT 'TESTCOVN' FUNC RET-CODE | 100 | 1 | 0 | 1 | x1 | C |
| 0 | | | 100 | INCLUDE TESTCOVC 'RET-CODE' 'FUNC' | 100 | 1 | 0 | 1 | x1 | C |
| 1 | COVDEMO | TESTCOVC | 10 | * Test Coverage | | 0 | 0 | 0 | | * |
| 1 | COVDEMO | TESTCOVC | 20 | * Copycode TESTCOVC | | 0 | 0 | 0 | | * |
| 1 | COVDEMO | TESTCOVC | 30 | IF &1& > 0 | 100 | 1 | 0 | 1 | x1 | C |
| 1 | COVDEMO | TESTCOVC | 40 | IF &1& = 1 | 0 | 0 | 1 | 1 | x0 | M |
| 1 | COVDEMO | TESTCOVC | 50 | PRINT 'Unsupported function' &2& | 0 | 0 | 1 | 1 | x0 | M |
| 1 | COVDEMO | TESTCOVC | 60 | ELSE | 0 | 0 | 1 | 1 | x0 | M |
| 1 | COVDEMO | TESTCOVC | 70 | PRINT 'Return code:' &1& | 0 | 0 | 1 | 1 | x0 | M |
| 1 | COVDEMO | TESTCOVC | 80 | END-IF | | 0 | 0 | 0 | | + |
| 1 | COVDEMO | TESTCOVC | 90 | END-IF | | 0 | 0 | 0 | | + |
| 0 | | | 110 | END-FOR | 100 | 1 | 0 | 1 | x1 | C |
| 0 | | | 120 | * | | 0 | 0 | 0 | | * |
| 0 | | | 130 | END | 100 | 1 | 0 | 1 | x1 | C |

Explanations:

- The source lines of the `TESTCOVC` copycode are included in the source of the `TESTCOVP` program and placed right after the corresponding `INCLUDE` statement.
- The lines 40 through 70 (in pink) of the copycode contain missed statements which means they were not executed in the test run.
- All other lines (in green) containing executable statements are covered.
- A `FOR` statement corresponds to four object code instructions. All four instructions are covered as indicated by `x1111` in the **Item Coverage** column.

## Statistics

If `STATISTICS=ON` is specified, the Profiler statistics are listed.

If the data is exported in CSV (comma-separated values) format, the properties and values of the Profiler statistics are added as separate columns to the event or consolidation trace. If coverage data is exported in CSV format, the statistics values are added in additional lines indicated by the value `Statistics` in the Evaluation column.

**Example of Statistics**

The following example shows an extract of the statistics of an NPRC resource file:

```
***********************************************************************
* 11:02:39          ***** NATURAL PROFILER UTILITY *****         2015-08-05
* User SAGTEST1                - Statistics -                       RESDATA
*
* General Info
* Machine class ...................... MAINFRAME
* Environment ........................ Batch
...
* Profiler Resource File
* Resource name ...................... EDM-MONITOR.nprc
* Resource type ...................... Natural Profiler Resource Consolidated
* Resource allocation date ........... 2015-07-27 10:36:19.6
* Resource size (bytes) .............. 565160
...
*
* Monitor Session
* Monitor start time ................. 2015-07-27 10:20:57.2
* Monitor end time ................... 2015-07-27 10:21:42.8
* Monitor elapsed time (sec) ......... 45.519604
*
* Trace Session
* First library ...................... SYSEDMD
* First program ...................... MENU
* Highest level ...................... 10
* Trace start time ................... 10:20:58.219911
* Trace end time ..................... 10:21:42.248348
* Trace elapsed time (sec) ........... 44.028437
...
* Data Processing
* Number of events ................... 895936
...
* Data Consolidation
* Consolidation ...................... ON
* Consolidation records .............. 21624
* Consolidation elapsed time (sec) ... 15.643516
* Consolidation factor ............... 41.4
* Consolidation records/block ........ 191.3
* Bytes/consolidation record ......... 25.8
*
***********************************************************************
```

Explanations:

- The `EDM-MONITOR.nprc` resource was allocated on `2015-07-27` at `10:36:19` a.m. and has a size of `565160` bytes.
- The profiled application was running on the same day at `10:20:58` a.m. for `44.0` seconds and started with the program `MENU` in the library `SYSEDMD`.
- The profiled application generated a total of `895936` Natural events. The data consolidation took `15.6` seconds and reduced the number of records to `21624` which corresponds to a consolidation factor of `41.4`.

All statistics information provided is explained in the section *Profiler Statistics*.

## Exporting Event Data for MashZone

You can visualize Profiler event data on an interactive MashZone dashboard by using the **Natural Profiler MashApp**.

The Profiler utility `MASHZONE` function reads the consolidated data of an NPRC resource file and writes the data to Work File 7 in the format expected by the Natural Profiler MashApp. The data of Work File 7 has to be exported by any tool (like FTP) as CSV (comma-separated values) file to the Natural Profiler directory in the MashZone environment before it can be accessed by the Natural Profiler MashApp.

Syntax of `MASHZONE`:

```
FUNCTION=MASHZONE
 [RESOURCE-NAME=resource-name]
 [RESOURCE-LIB=library-name]
```

Syntax Description:

| Keyword | Value | Description |
|---|---|---|
| `RESOURCE-NAME` | *`resource-name`* | The name of the Natural Profiler resource consolidated (NPRC) file to be exported for MashZone.<br><br>The extension `.nprc` is added automatically.<br><br>Default: The name of the last created NPRC resource file in the library |
| `RESOURCE-LIB` | *`library-name`* | The name of the Natural library that contains the NPRC resource file you want to export.<br><br>Default: The name of the current library |

### Alternative Function Specifications

**READ**

The following Profiler utility `READ` function is equivalent to the `MASHZONE` function and generates the same export data:

```
FUNCTION=READ
  RESOURCE-NAME=resource-name
  RESOURCE-LIB=library-name
  RESOURCE-TYPE=NPRC
  EVENT=ON
  PROGRAM=OFF
  STATISTICS=ON
  PRINT=OFF
  EXPORT=ON
  FORMAT=SEMICOLON
```

**CONSOLIDATE**

The **Natural Profiler MashApp** can also process data exported with the Profiler utility `CONSOLIDATE` function if you specify the following keywords:

```
FUNCTION=CONSOLIDATE    /* Consolidate Profiler data
  EXPORT=ON             /* Write to Work File 7
  FORMAT=SEMICOLON      /* CVS format with semicolon separator
  ...
```

### Example of MASHZONE

The following example reads the consolidated Profiler resource `Test.nprc` in the library `PRFDATA`. The data is written in CSV (comma-separated values) format to Work File 7 which can be exported to MashZone.

```
FUNCTION=MASHZONE       /* Export Profiler data for MashZone
  RESOURCE-NAME='Test'  /* Resource name
  RESOURCE-LIB=PRFDATA  /* Resource library
```

## Maintaining Profiler Resource Files

In general, Profiler resources are listed as NPRF, NPRC or NCVF files by using the Natural SYSMAIN utility, NaturalONE or Natural Studio. These tools also provide functions to copy, rename and delete resource files.

In addition, you can use Profiler utility functions to list and delete Profiler resource files.

This section covers the following topics:

- Listing Profiler Resource Files

- Deleting a Resource File

## Listing Profiler Resource Files

The Profiler utility `LIST` function lists the Profiler resource files of a given Natural library and the date and time when the resource files were allocated.

Syntax of `LIST`:

```
FUNCTION=LIST
 [RESOURCE-LIB=library-name]
 [RESOURCE-TYPE={NPRF|NPRC|NCVF}]
 [PRINT={ON|OFF}]
 [EXPORT={ON|OFF}]
 [FORMAT={TEXT|COMMA|SEMICOLON}]
```

Syntax Description:

| Keyword for LIST | Value | Description |
|---|---|---|
| RESOURCE-LIB | *library-name* | The name of the Natural library that contains the Profiler resource files you want to list.<br><br>Default: The name of the current library |
| RESOURCE-TYPE | | Specifies the type of resource files to be listed: NPRF, NPRC or NCVF.<br><br>Default: All types are listed if no value is specified here. |
| | NPRF | List NPRF (Natural Profiler Resource File) resource files only. |
| | NPRC | List NPRC (Natural Profiler Resource Consolidated) resource files only. |
| | NCVF | List NCVF (Natural code coverage file) resource files only. |
| PRINT | | Specifies whether the result is written to standard output. |
| | ON | Write to standard output. |
| | OFF | Do not write to standard output. |
| EXPORT | | Specifies whether the result is written to Natural Work File 7. |
| | ON | Write to Work File 7. |
| | OFF | Do not write to Work File 7. |
| FORMAT | | Specifies the format in which the exported data is written to Work File 7. |
| | TEXT | Write the data in free text format. |
| | COMMA | Write the data in CSV format with a comma (,) used as a separator. |
| | SEMICOLON | Write the data in CSV format with a semicolon (;) used as a separator. |

**Example of LIST**

The following example lists the NPRF Profiler resource files of library PRFDATA. The list is written to standard output and to Work File 7 in text format.

```
FUNCTION=LIST           /* List Profiler resource files
  RESOURCE-LIB=PRFDATA  /* Resource library
  RESOURCE-TYPE=NPRF    /* List NPRF resource files
  PRINT=ON              /* Write to standard output
  EXPORT=ON             /* Write to Work File 7
  FORMAT=TEXT           /* Export in text format
```

Output:

```
Natural Profiler Resources
--------------------------
Library: PRFDATA
Resource type: nprf

Count Date       Time     Name
    1 2015-06-15 14:32:18 Hello1.nprf
    2 2015-06-26 18:39:57 QDTest1.nprf
    3 2015-06-24 22:00:35 QETest1.nprf
    4 2015-06-30 14:32:42 Studio.nprf
    5 2015-07-02 15:02:32 Test.nprf

Number of nprf resources in library PRFDATA: 5
```

### Deleting a Resource File

If you delete or replace a big resource file, it can happen that you receive the following error message:

```
Error - NAT3047 Maximum value for Adabas parameter NISNHQ was exceeded.
```

In this case, you have two options:

1. Contact your database administrator to increase the Adabas parameter NISNHQ.
2. Use the Profiler utility DELETE function to perform a “dirty” delete of the resource. This function does not delete the resource in one big step but in chunks (with an end of transaction after each chunk). If the DELETE function fails by any reason, you need to repeat it to get rid of inconsistent data.

   Syntax of DELETE:

```
FUNCTION=DELETE
 [RESOURCE-NAME=resource-name]
 [RESOURCE-TYPE={NPRF|NPRC|NCVF|NPRK|NONE}]
 [RESOURCE-LIB=library-name]
```

Syntax Description:

| Keyword for DELETE | Value | Description |
|---|---|---|
| RESOURCE-NAME | *resource-name* | The name of the Profiler resource file you want to delete.<br><br>Possible extensions are .nprf (Natural Profiler resource file), .nprc (Natural Profiler resource consolidated), ), .ncvf (Natural code coverage file) or .nprk (Natural Profiler resource key). If no extension has been specified, the extension specified with the keyword RESOURCE-TYPE is added automatically.<br><br>Default: none<br><br>If RESOURCE-TYPE=NPRK is specified and no RESOURCE-NAME, the name of the Profiler resource file containing the NaturalONE Profiler key is used as *resource-name*. |
| RESOURCE-TYPE | | The default resource type (extension) to be deleted if no extension is specified with RESOURCE-NAME.<br><br>Default: NPRF |
| | NPRF | The default resource type is NPRF with the extension .nprf. |
| | NPRC | The default resource type is NPRC with the extension .nprc. |
| | NCVF | The default resource type is NCVF with the extension .ncvf. |
| | NPRK | The default resource type is NPRK with the extension .nprk. |
| | NONE | The resource with the short name *resource-name* is deleted. |
| RESOURCE-LIB | *library-name* | The name of the Natural library that contains the resource you want to delete.<br><br>Default: The name of the current library |

**Example of DELETE**

```
FUNCTION=DELETE         /* Delete a Profiler resource file
  RESOURCE-NAME='Test'  /* Resource name
  RESOURCE-TYPE=NPRF    /* Resource type
  RESOURCE-LIB=PRFDATA  /* Resource library
```

## Including Profiler Input from Natural Text Objects

The Profiler can read input data from a Natural text object. The syntax of the data in the Natural text object is the same as for the primary command input data set CMSYNIN (see ***Syntax and Keywords***).

**To include Profiler input data from a Natural text object**

- Enter the following Profiler keywords:

  ```
  INCLUDE-LIB=library-name
  INCLUDE=object-name
  ```

  The keyword syntax is explained in ***Profiler Utility Keywords***.

The data in the Natural text object is added to the Profiler input data in the line after the `INCLUDE` keyword. The Profiler input data can contain multiple `INCLUDE` keywords, and the related Natural text objects can also contain `INCLUDE` keywords. If a Natural text object contains an `END-PROFILER` keyword, the Profiler utility terminates and any remaining data in the Natural text object(s) is ignored.

The Natural system library `SYSPRFLR` supplies text object whose names begin with `X` which can be used as Profiler input. The individual Profiler functions they perform are described in the sources of these objects.

We recommend that you do not modify any objects in the system library `SYSPRFLR` because they can be overwritten or removed when a new Natural version is installed. Copy the required object(s) to a user library before you edit it.

### Examples of INCLUDE

The following example adds the contents of the Natural text object `MYPROF` from the library `MYLIB` to the Profiler input data:

```
INCLUDE-LIB=MYLIB
INCLUDE=MYPROF
```

The following example adds the contents of the Natural text object `XINIT` from the library `SYSPRFLR` to the Profiler input data. The object initializes and starts profiling without consolidation and without statement event collection. Additionally, it terminates the Profiler utility so that no further Profiler input is expected after the `INCLUDE` keyword.

```
INCLUDE=XINIT
```

## Event Trace

The Natural Profiler collects detailed information of each Natural event that occurs while a Natural application executes. This data can be viewed in the event trace.

The traces written for Natural code coverage are described in the section *Tracing Natural Code Coverage*.

The Profiler utility provides the following options to write a Profiler event trace:

- Write the trace to standard output of the Profiler monitor session (MONPRINT data set) while the application is profiled.
- Write the trace to standard output while the NPRF data is consolidated. In this case, the event trace shows the delta values of the elapsed time and the CPU time instead of event-specific data.
- Write the trace when reading a Profiler NPRF resource file with the Profiler utility `READ` function.

**Note:** The event trace can also be listed in NaturalONE.

#### To enable the event trace

- Enter the following subordinate keyword of the Profiler utility `INIT` function:

```
TRACE-EVENT=ON
```

Enter the following subordinate keyword of the Profiler utility `CONSOLIDATE` function:

```
TRACE-EVENT=ON
```

Enter the following subordinate keyword of the Profiler utility `READ` function:

```
EVENT=ON
```

The Profiler event trace contains the following columns:

| Column | Description |
|---|---|
| Count | Event count |
| Time | Event time<br><br>Unit: *hour*:*minute*:*second*.*microseconds* |
| CPU-Time | Session CPU time<br><br>Unit: microseconds |
| Ev | Event type; see ***Events and Data Collected***. |
| Lev | Program level |
| Library | Program library |
| Program | Program (Natural object) name |
| Line | Line number of program statement executed |
| CC-Lib | Copycode library (if copycode is active) |
| CC-Name | Copycode name (if copycode is active) |
| Statement | Natural statement currently executed. For technical reasons, there is no one-to-one relationship between a Natural source code statement and the corresponding object code in the cataloged object. Therefore, the statements listed in the Profiler event trace can differ from the statements in the source. |
| Local-Data | Event-specific data like the Adabas database ID (DBID) and file number (FNR).<br><br>This data is only displayed for the Profiler utility `INIT` and `READ` functions. |
| Elapsed (ms) | Elapsed time spent processing the event.<br><br>Unit: milliseconds<br>This data is only displayed for the Profiler utility `CONSOLIDATE` function. |
| CPU-Delta | CPU time spent processing the event.<br><br>Unit: milliseconds |

### Example of an Event Trace

In the following example, the Profiler utility READ function prints the event trace:

```
FUNCTION=READ                /* Read event data
  EVENT=ON                   /* Write event trace
```

The event trace is written to standard output:

```
Count Time              CPU-Time (ms) Ev Lev Library  Program  Line CC-Lib   CC-Name  Statement  Local-Data
    0 17:38:17.200951         42.324 MP 003 SYSPRFLR PRBINIT  8370                 Call       Monitor pause requested
    0 17:38:17.204508         43.471 MP 003 SYSPRFLR PRBSTART 1760                   Call       Start of block filter
   11 17:38:17.218379         48.874 DB 000                   0000                               00010/00032 S1
   12 17:38:17.218941         48.897 DA 000                   0000                               00010/00032 S1   Rsp: 0
   13 17:38:17.218944         48.910 PL 000                   0000                               Execute PRFDEMO/XPROF
   14 17:38:17.218945         48.916 PS 001 PRFDEMO  XPROF    0000                   PgmStart   00010/00032 Type: P
   15 17:38:17.218956         48.979 IB 001 PRFDEMO  XPROF    0300                   Input      Out: 133 In: 0
   16 17:38:17.219235         49.046 IA 001 PRFDEMO  XPROF    0300                   Input      Out: 133 In: 80
   17 17:38:17.219258         49.182 DB 001 PRFDEMO  XPROF    0370                   Callnat    00010/00032 S1
   18 17:38:17.220426         49.211 DA 001 PRFDEMO  XPROF    0370                 Callnat    00010/00032 S1   Rsp: 0
   19 17:38:17.220427         49.216 DB 001 PRFDEMO  XPROF    0370                   Callnat    00010/00032 S1
   ...
```

## Tracing Natural Code Coverage

When **Natural code coverage** is performed, the executed events can be viewed in the **event trace**.

When the Profiler reads a generated program (GP), the event trace also shows the statements contained in the GP. The data which is written to the Natural coverage resource file can be listed in the coverage trace. When the resource is read with the Profiler utility READ function, the coverage data can be traced with the internal data trace.

#### To enable tracing for code coverage

1. Enable the event trace as described in ***To enable the event trace*** in the section *Event Trace*.
2. Enable the coverage trace by specifying the following subordinate keyword of the Profiler utility COVERAGE function:

   ```
   TRACE-COVERAGE=ON
   ```

3. Enable the internal trace by specifying the following subordinate keyword of the Profiler utility READ function:

```
TRACE=9
```

The table below describes the properties listed in the traces and indicates with (X) for which type of trace the data is provided:

| Property | Event Trace | Coverage Trace | Internal Trace | Description |
|---|---|---|---|---|
| Count | X | X | X | The event count. |
| Ev | X | X | X | The event type.<br><br>See *Events and Data Collected*. |
| Library | X | X | X | The name of the Natural library that contains the program/object. |
| Program/Object | X | X | X | The name of the Natural program/object. |
| Ty | - | X | X | The object type such as `P` for program. |
| CC-Lib | X | X | X | The name of the Natural library that contains the copycode (if copycode is active). |
| CC-Name | X | X | X | The name of the copycode. |
| Line | X | X | X | The source line number. |
| Statement | X | X | - | The Natural statement executed.<br><br>For technical reasons, there is no one-to-one relationship between a Natural source code statement and the corresponding object code in the cataloged object. Therefore, the statements listed in the Profiler event trace can differ from the statements in the source.<br><br>For code coverage, the statement object code is not saved in the resource file. Therefore, it can only be listed during data collection. |
| GP-Offset | X | - | - | The offset in the GP.<br><br>It uniquely identifies the statement item at execution time. |
| Size | X | - | - | The size of the statement in the GP. |
| CC-ID | - | X | X | The copycode ID.<br><br>It uniquely identifies the copycode instance in the GP. The program gets the copycode ID `0`. |
| Par-CC | - | X | X | For copycode only.<br><br>The parent copycode ID which is the copycode ID of the object/copycode that includes the current copycode. |

| Property | Event Trace | Coverage Trace | Internal Trace | Description |
|---|---|---|---|---|
| INCL | - | X | - | For copycode only.<br><br>The line number of the `INCLUDE` statement that includes the copycode. |
| FirstS | - | X | X | The ID of the first statement of the object or copycode. |
| Stmts | - | X | X | The total number of executable statements in the object whereby all `INCLUDE` statements are resolved. |
| Item | X | X | X | The item ID of the statement.<br><br>It uniquely identifies the statement in the resource file. |
| Cover | X | X | X | The coverage flag (`0` or `1`) of the statement.<br><br>When the GP is read, all flags are initialized with `0`. Whenever a statement is executed, the flag is set to `1`. |

**Example of a Coverage Trace**

In the following example, the Profiler utility `COVERAGE` function writes the coverage trace:

```
FUNCTION=COVERAGE           /* Initialize code coverage
  TRACE-COVERAGE=ON         /* Write coverage trace
```

The coverage trace is written to the standard output of the Profiler monitor session (MONPRINT data set):

```
                        Natural Coverage Trace                                  ↩

                        ----------------------                                  ↩

Count Ev Library  Program  Ty CC-Lib   CC-Name  Line Statement  CC-ID Par-CC INCL ↩
 FirstS  Stmts    Item Cover
    1 PI COVDEMO  TESTCOVN N                                        0      0 0000 ↩
      1     44         1
    2 NS COVDEMO  TESTCOVN                      0130 Compute        0            ↩
                     4 0
    3 NS COVDEMO  TESTCOVN                      0140 Print          0            ↩
                     5 0
    4 NS COVDEMO  TESTCOVN                      0150 Goto           0            ↩
                     6 0
    5 NS COVDEMO  TESTCOVN                      0310 If             0            ↩
                    39 0
    6 NS COVDEMO  TESTCOVN                      0310 Compute        0            ↩
                    40 0
    7 NS COVDEMO  TESTCOVN                      0320 Print          0            ↩
                    41 0
    8 NS COVDEMO  TESTCOVN                      0340 Compute        0            ↩
                    43 0
```

```
    9 NS COVDEMO  TESTCOVN                      0100 Reset          0             ↩
                     1 1
   10 NS COVDEMO  TESTCOVN                      0120 Reset          0             ↩
                     2 1
   11 NS COVDEMO  TESTCOVN                      0130 If             0             ↩
                     3 1
   12 NS COVDEMO  TESTCOVN                      0150 If             0             ↩
                     7 1
...
```

## Internal Trace

The Profiler internal trace writes Profiler messages such as errors or warnings.

The internal trace can be activated for the following:

- The Profiler monitor sessions (**data collection**). The data is written to the standard output of the monitor session.
- The Profiler data processing functions. The data is written to standard output.

### To activate the internal trace for the Profiler trace session or the data processing functions

- Enter the following Profiler keyword:

  ```
  TRACE=n
  ```

  where *n* is the trace level (see ***Trace Levels***).

  **Notes:**

  1. By default (if `TRACE` is not specified), Trace Level 2 (warnings) is used.
  2. The trace is activated as soon as the `TRACE` keyword is specified. It is therefore recommended to specify the `TRACE` keyword as soon as possible.
  3. If you execute the Profiler utility multiple times in the job, you need to specify the `TRACE` keyword with each execution.

### To activate the internal trace for the Profiler monitor session

- Enter the following subordinate keyword of the Profiler utility `INIT` or `COVERAGE` function:

```
TRACE-MONITOR=n
```

where *n* is the trace level (see ***Trace Levels***).

**Note:** By default (if `TRACE-MONITOR` is not specified), Trace Level 3 (statistics) is used.

## Trace Levels

The trace levels used by the Profiler trace and monitor sessions and by the Profiler data processing functions are listed in the following table. In general, a higher trace level also contains the information of the lower trace levels. For example, if you select Trace Level 3 (statistics), error messages and warnings are also logged.

We recommend that you use at least Trace Level 2 (warnings) so that error messages and warnings are logged. For the Profiler monitor session, Trace Level 3 (statistics) is a good choice. It prints the statistics of the Profiler run (see ***Profiler Statistics***). Higher trace levels for the monitor session can be extremely verbose and the output can be mixed up with the event trace (if activated).

| Trace Level | Name | Description |
|---|---|---|
| 0 | No trace | Profiler internal trace is deactivated. |
| 1 | Error | Log error messages. |
| 2 | Warning | Log warnings. |
| 3 | Statistics | Trace session: Print the values used for the Profiler utility `INIT` or `COVERAGE` function.<br><br>Monitor session: Print the ***profiler statistics***.<br><br>Data consolidation: Print the ***profiler statistics*** including the consolidation statistics. |
| 4 | Function | Log messages for used **Profiler utility keywords** (`FUNCTION`, `FILTER`, etc.). |
| 5 | Block | Print the statistics of each data block written to the Profiler resource file. |
| 6 | Details | Log detailed information. |
| 7 | | Not used. |
| 8 | | Not used. |
| 9 | Data | Trace the coverage resource data when reading an NCVF coverage resource file. |
| 10 | Internal | Internal usage. |

### Example

In the following example, the Profiler internal trace is set to `4` (function) for the trace session:

```
* Set Profiler internal trace
TRACE=4                     /* Trace level
```

Output of the Profiler trace session for Trace Level 4:

```
PRBMAIN : Profiler trace level: 4
PRBMAIN : Profiler On-Error: Terminate
************************************************************************
* 15:45:14          ***** NATURAL PROFILER UTILITY *****         2014-12-17
* User SAGPRFD1               - Function INIT -                     PRBINIT
*
* Keyword              Value
* -------------------- ----------------------------------------------------
* Resource             ON
* Resource-Lib         SAGPROF
* Resource-Name        Test01.nprf
* Replace              Y
* Wait-Full            60
* Wait-Empty           60
* Sampling             OFF
* Consolidate          OFF
* Trace-Monitor        5
* Trace-Event          OFF
* Trace-Consolidate    OFF
* -------------------- ----------------------------------------------------
************************************************************************
PRBINIT : Profiler INIT function - Start monitor session.
PRBINIT : Profiler INIT function - Monitor session started. Time: 1.0 sec.
PRBINIT : Set trace session Id ...: 0000000000000001
PRBINIT : Set monitor session Id .: 0000000100000000
PRBINIT : Trace session successfully initialized.
PRBFEVEN: Event filter: SI ST DB DA PL PS PT PR IB IA E CB CA U RS RI RO NS
PRBSTART: Profiling started.
PRBMAIN : Profiler - End of input.
************************************************************************
...
Output of the application
...
```

## Profiler Statistics

In addition to event data, the Profiler collects statistical data which is written to the Profiler resource file.

The Profiler utility provides the following options to write and view Profiler statistics:

- Write the statistics to the standard output of the Profiler monitor session (MONPRINT data set) while the application is profiled or code coverage is performed.

- Write the statistics to standard output while the data is consolidated.
- Write the statistics when reading a Profiler resource file with the Profiler utility `READ` function.
- View the statistics with the **Natural Profiler MashApp**.

**To write Profiler statistics, perform one of the following steps**

- Enter the following subordinate keyword of the Profiler utility `INIT` or `COVERAGE` function:

  ```
  TRACE-MONITOR=3
  ```

  or a higher trace level (see ***Trace Levels***). Trace Level 3 is also the default level for the Profiler monitor session.
- Enter the following keyword before you start the Profiler utility `CONSOLIDATE` function:

  ```
  TRACE=3
  ```

  or a higher trace level (see ***Trace Levels***).
- Enter the following subordinate keyword of the Profiler utility `READ` function:

  ```
  STATISTICS=ON
  ```

The Profiler statistical data is displayed in categories combining properties of a similar type. The following categories are available:

- General Info
- Profiler Resource File
- Monitor Session
- Trace Session
- Data Processing
- Event Type Statistics
- Monitor Pause Statistics
- Data Consolidation
- Coverage

**Note:** The properties listed in the following section are the properties provided by the Profiler in all environments. The Profiler Statistics contains only the properties that are relevant for the current run. Therefore, not all of the properties listed in the section are displayed in every case.

### General Info

Display environment and Natural Profiler related information.

| Property | Unit | Description |
|---|---|---|
| Machine class | | The name of the machine class on which the Natural application is running. |
| Environment | | The environment in which the Natural application is running, such as `NaturalONE`, `Batch` or `RPC`. |
| Codepage | | The code page used while the Natural application was monitored. |
| User | | The ID of the user running the application (value of `*USER`). For a batch job, it can contain the name of the job. |
| Profiler version | | The internal version of the Profiler.<br><br>NaturalONE environment: The version of the Profiler on the server. |
| Profiler revision | *`vvrr.xxx`* | The Profiler revision is build up by the Natural version and the last Profiler correction number. |
| Profiler revision date | *`yyyy-mm-dd hh:ii`* | The date and time when the Profiler revision was created. |
| Profiler client version | | NaturalONE environment: The version of the Profiler client. |
| Profiler trace library | | NaturalONE environment: The name of the Natural library containing the Profiler internal trace and the Profiler event trace. |
| Profiler trace level | | The level of the Profiler internal trace. |
| Profiler trace member | | NaturalONE environment: The name of the Natural text object containing the Profiler internal trace. |
| Profiler event trace | | Indicates whether the Profiler event trace was activated (`ON`/`OFF`). |
| Profiler event trace member | | NaturalONE environment: The name of the Natural text object containing the Profiler event trace. |
| Utility trace level | | NaturalONE environment: The Natural utilities trace level. |

### Profiler Resource File

Display Profiler resource file related information.

| Property | Unit | Description |
|---|---|---|
| Resource name | | The name of the Natural Profiler resource file. |
| Resource type | | The type of the Natural Profiler resource file: Natural Profiler resource file (NPRF), Natural Profiler resource consolidated (NPRC) or Natural code coverage file (NCVF). |
| Resource short name | | Mainframe: The short name of the Natural Profiler resource file. |
| Resource library | | The name of the Natural library containing the Natural Profiler resource file. |
| Resource DBID | | The database ID of the Natural library containing the Natural Profiler resource file. |
| Resource FNR | | The file number of the Natural library containing the Natural Profiler resource file. |
| Resource allocation date | `yyyy-mm-dd hh:ii:ss.t` | The date and time when the Natural Profiler resource file was allocated. |
| Resource size | bytes | The size of the Natural Profiler resource file. It comprises the resource headers, the event data and the properties. The resource size is calculated regardless whether the resource is allocated or not. |
| Resource block size | bytes | The maximum size of a resource block. A resource block consists of a resource block header and a data block. |
| Resource version | | The version of the Natural Profiler resource layout. |

### Monitor Session

Display statistics of the Profiler monitor session.

| Property | Unit | Description |
|---|---|---|
| Monitor start time | `yyyy-mm-dd hh:ii:ss.t` | The date and time when the monitor session started. |
| Monitor end time | `yyyy-mm-dd hh:ii:ss.t` | The date and time when the monitor session ended. |
| Monitor elapsed time | sec | The total elapsed time consumed by the monitor session. |

### Trace Session

Display statistics of the Profiler trace session. The Profiler trace session includes also the application execution.

| Property | Unit | Description |
|---|---|---|
| First library | | The first library monitored. The libraries `SYSTEM`, `SYSLIB*` and `SYSPRF*` are ignored. |
| First program | | The first program monitored. |
| Highest level | | Highest level number of the Natural objects monitored. |
| Trace start time | *`hh:ii:ss.microsec`* | The start time of the tracing. With NaturalONE this is the time of the `SI` (session initialization) event. In batch, the session is already initialized when the monitoring starts. Therefore, the start time is the time of the first event (usually a Monitor Pause event). |
| Trace end time | *`hh:ii:ss.microsec`* | The end time of the tracing. This is in general the time of the `ST` (session termination) event. |
| Trace elapsed time | sec | The elapsed time consumed by the trace session from the start time to the end time. |
| Application CPU time | ms | The total CPU time consumed by the application. |
| Monitor CPU time | ms | The total CPU time consumed by the Natural data collector. This time is not measured by the Natural UNIX or Windows server. |
| Total CPU time | ms | The total CPU time consumed by the trace session. It is the sum of the application CPU time and the monitor CPU time. |
| Sampling interval | microsec | The sampling interval time (CPU time in microseconds). A value of zero (`0`) means that no sampling was active. |
| Data pool empty | | The number of Profiler read requests which found the Profiler data pool empty (and a session active). |
| Data pool empty after full | | The number of Profiler read requests which found the Profiler data pool empty although it was full before. If this counter is greater than `0`, the Profiler data pool is too small which leads to a poor performance. |
| Data pool overflow | | The number of Profiler data pool overflows (with data lost). Data pool overflows should no longer happen. This property is only maintained for backward compatibility with previous versions of Natural. |
| No session active | | The number of read requests which found the Profiler data pool empty and no trace session active. This can only happen for Profiler read requests submitted before the session initialization or after the session termination. |

### Data Processing

Display statistics of the data processing, compression and transfer.

| Property | Unit | Description |
|---|---|---|
| Number of events | | The total number of events. |
| Highest event number | | The highest event number as given by the Natural data collector. Note that the Natural data collector counts only non-statement events when called from NaturalONE. In batch it depends on the statement filter whether statement events are counted or not. |
| Number of data blocks | | The number of event data blocks send to NaturalONE or written to the resource. |
| Utility buffer size | bytes | The size of the utility buffer used for the data transfer from the server to NaturalONE. In general, the buffer contains the header information and function-specific data. |
| Data block size | bytes | The maximum amount of event data which can be transferred from the server to NaturalONE in one call. The same data block size is used for storing the event data in the resource file. |
| RDC data length | bytes | The total size of the data received from the Natural Data Collector. |
| Uncompressed data length | bytes | The total size of the Profiler data in uncompressed format. |
| Compressed data length | bytes | The total size of the compressed data as send to NaturalONE or written to the Profiler resource file. |
| Identical bytes trimmed left | | The number of identical bytes trimmed left at the forward data compression. |
| Blanks trimmed right | | The number of blanks trimmed right at the backward data compression. |
| Compression header length | bytes | The total size of the compression headers saved with each compressed event record. |
| Compression rate | percent | The percentage of the data reduction by the compression. The higher the compression rate, the less data has to be transferred or saved. The formula of the compression rate is described below. |
| Events/block | | The average number of events contained in one event data block. |
| Bytes/event | | The average length in bytes of a compressed event data record. This property is not available for consolidated or coverage data. |

The compression rate is calculated by the following formula:

$$\text{CompressionRate} := 100 \times \frac{\text{BytesTrimmedLeft} + \text{BytesTrimmedRight} - \text{CompressionHeaderLength}}{\text{UncompressedDataLength}}$$

### Event Type Statistics

Display statistics of the event types.

| Property | Description |
|---|---|
| Unknown event | The number of unknown events. |
| Session initialization | The number of Session Initialization events. |
| Session termination | The number of Session Termination events. |
| Program load | The number of Program Load events. |
| Program start | The number of Program Start events. |
| Program termination | The number of Program Termination events. |
| Program resume | The number of Program Resume events. |
| Program information | The number of Program Information events. |
| Before database call | The number of Before Database Call events. |
| After database call | The number of After Database Call events. |
| Before terminal I/O | The number of Before Terminal I/O events. |
| After terminal I/O | The number of After Terminal I/O events. |
| Before external program call | The number of Before External Program Call events. |
| After external program call | The number of After External Program Call events. |
| Runtime error | The number of Runtime Error events. |
| Natural statement | The number of Natural Statement events. For technical reasons, multiple Natural statements can be merged into one statement event and conversely, one Natural statement can cover multiple statement events. |
| Outbound RPC message | The number of Outbound RPC Message events. |
| Inbound RPC message | The number of Inbound RPC Message events. |
| Start RPC request execution | The number of Start of RPC Request Execution events. |
| RPC Wait for Client | The number of RPC Wait for Client events. |
| User trace call | The number of User-Defined Events. |
| Monitor pause | The number of Monitor Pause events. |
| Monitor filter | The number of monitor filter events. Filter events are not recorded. |

### Monitor Pause Statistics

Display statistics of the types of Monitor Pause events.

| Property | Description |
|---|---|
| Pause - unknown type | The number of Monitor Pause events with unknown pause type. |
| Pause - requested | The number of requested Monitor Pause events. |
| Pause - start of block filter | The number of Monitor Pause events caused by a start of a block filter (library, program, line, FNAT, event count or time filter). |
| Pause - data pool full | The number of Monitor Pause events caused by a data pool full situation. |
| Pause - data pool overflow | The number of Monitor Pause events caused by a data pool overflow situation. |

### Data Consolidation

Display statistics of the data consolidation.

| Property | Unit | Description |
|---|---|---|
| Consolidation | | Indicates whether the Profiler data is consolidated (`ON`/`OFF`). The consolidation aggregates similar events into one consolidation record. |
| Consolidation records | | The total number of consolidation records. In general, a consolidation record comprises multiple events. |
| Consolidation elapsed time | sec | The elapsed time in seconds required for the data consolidation with the Profiler utility `CONSOLIDATE` function.<br><br>This value is not provided when the consolidation is performed during data collection (Profiler utility `INIT` function). In this case, the time required for the consolidation is contained in the **Monitor elapsed time**. |
| Consolidation factor | | The average number of events combined into one consolidation record. The higher the consolidation factor, the better the consolidation.<br><br>`ConsolidationFactor := NumberOfEvents / ↩ ConsolidationRecords` |
| Consolidation records/block | | The average number of consolidation records contained in one data block. |
| Bytes/consolidation record | | The average length in bytes of a compressed consolidation record. |
| Consolidate I/O time | | Indicates whether I/O and Natural RPC client time are included in the consolidated data. |

### Coverage

Display statistics of Natural code coverage.

**Note:** Natural code coverage statistics are collected on the mainframe only.

| Property | Description |
|---|---|
| Coverage | Indicates whether Natural code coverage is performed (`ON/OFF`). |
| Missed statements recorded | Indicates whether missed statements are recorded (`ON/OFF`). |
| Coverage records | The total number of coverage records. These are program information and Natural statement records. |
| Program information records | The number of program information records written to the resource file. Each program information record contains program and copycode related information. |
| Coverage records/block | The average number of coverage records contained in one data block. |
| Bytes/coverage record | The average length in bytes of a compressed coverage record. |
| Programs covered | The number of covered programs. |
| Statement coverage | The percentage of statements of all accessed programs that have been covered by the application. |
| Statements covered | The number of covered (executed) statements. |
| Statements total | The total number of executable statements of all programs accessed. |

# 106 Natural Profiler MashApp


- Preparing to Use the MashApps ..... 894
- Preparing the Profiler Data ..... 898
- Opening the MashApps ..... 899
- Evaluation Page ..... 900
- Compare Page ..... 909
- Properties Page ..... 911
- Use Cases ..... 913

MashZone is a browser-based application from Software AG which is used to visualize data on a graphical, interactive dashboard, a so-called MashApp. The Natural Profiler MashApps evaluate the Profiler event data and depict it in MashZone.

## Preparing to Use the MashApps

This section provides instructions for implementing the MashApps:

- Downloading the MashApps
- Unpacking the Zip File
- Editing the Overview.csv Resource File
- Activating the MashApps

### Downloading the MashApps

The Natural Profiler MashApps and related data are supplied as a Natural component in a zip file.

#### To download the MashApps zip file

1. Log in to Software AG's Empower web site at ***https://empower.softwareag.com/*** (password required).
2. Go to **Products & Documentation** > **Download Components**.

   The **Download Components** section is displayed.
3. From the **Download Components** section, select **Natural Profiler MashApp**.
4. Download the `NaturalProfiler_MashApp.zip` file.

   In addition to the zip file, Empower also provides the Readme file `Readme_NaturalProfiler_MashApp.txt` which contains the latest update information.

### Unpacking the Zip File

You have to unpack the MashApp zip file in the appropriate MashZone directory which depends on the MashZone version installed at your site.

#### To unpack the MashApp zip file

- Unpack the `NaturalProfiler_MashApp.zip` file in the appropriate user data directory of MashZone:
  - For MashZone Version 9.0 and above:

```
installation-directory\server\bin\work\work_mashzone_server-type\mashzone_data
```

where *server-type* indicates the type of the MashZone server: `s`, `m` or `i`. For example, `work_mashzone_m` for a medium type.

- For MashZone versions below Version 9.0:

```
installation-directory
```

where *installation-directory* is the MashZone installation directory.

After unpacking the zip file, the following subdirectories are available in the user data directory of MashZone:

| Directory | Content |
|---|---|
| `importexport\Profiler_date` | MashApps for the Natural Profiler.<br><br>*date* is the MashApp generation date. |
| `resources\Profiler` | Parent directory of Profiler resources.<br><br>Contains the user-modified `Overview.csv` resource file.<br><br>See also ***Editing the Overview.csv Resource File***. |
| `resources\Profiler\Definition` | Resources used by the MashApp.<br><br>Initially, this directory contains the resources which do not have to be edited. |
| `resources\Profiler\Data` | Profiler data directory (including subdirectories) in which the Profiler data files are stored by default. |
| `resources\Profiler_src` | Source directory for resources which have to be edited and copied into the `resources\Profiler` directory.<br><br>See also ***Editing the Overview.csv Resource File***. |
| `assets\colorschemes` | Color schemes.<br><br>The color schemes for the Natural Profiler are named `Profiler_*.xml`. |

### Editing the Overview.csv Resource File

You can edit the `Overview.csv` resource file in the `resources\Profiler_src` directory to adapt the Natural Profiler MashApps to your requirements. The resource file is a CSV-formatted file with semicolon (`;`) separators which can be edited with any text editor.

The supplied `Overview.csv` file contains one line for the sample Profiler data in the `Profiler_Sample.csv` file in the `resources\Profiler\Data` directory. Add more lines for each Profiler CSV file you want to evaluate. For information on creating Profiler CSV files, see ***Preparing the Profiler Data***. You can also add or delete lines in the `Overview.csv` file later, after you have copied it to the `resources\Profiler` directory (see ***Activating the MashApp***).

In the columns of the `Overview.csv`, you can specify the following:

| Column | Description |
|---|---|
| `csv File` | Specify the name of the Profiler consolidated data file.<br><br>If the data file resides in a subdirectory of `resources\Profiler`, specify the relative path and the file name. For example:<br><br>Specify `Data\ProfilerTrace.csv` if the `ProfilerTrace.csv` data file is contained in `...\resources\Profiler\Data`. |
| `Description` | Specify a descriptive name for the Profiler consolidated data file. The descriptions are used in the **Input** selection box of the Natural Profiler MashApps.<br><br>If you do not enter a value, the value of the **csv File** column is used in the **Input** selection box. |
| `Enable` | If you enter `Y` in this column, the name or description of the Profiler consolidated data file is shown in the **Input** selection box. Otherwise, it is not shown. |

### Activating the MashApps

Prerequisites for activating the Natural Profiler MashApps are a Professional, Enterprise or Event license file and administrator rights.

#### To activate the MashApps

1. Copy the resource file from `resources\Profiler_src` to `resources\Profiler`.
2. Invoke MashZone.
3. Go to the **Administration** page (see the corresponding tab at the top of the page) and then to the **Import/Export/Delete** page.
4. Import the MashZone archive files (`*.mzp`) from the `importexport\Profiler_`*`date`* directory by using the **Import** function.

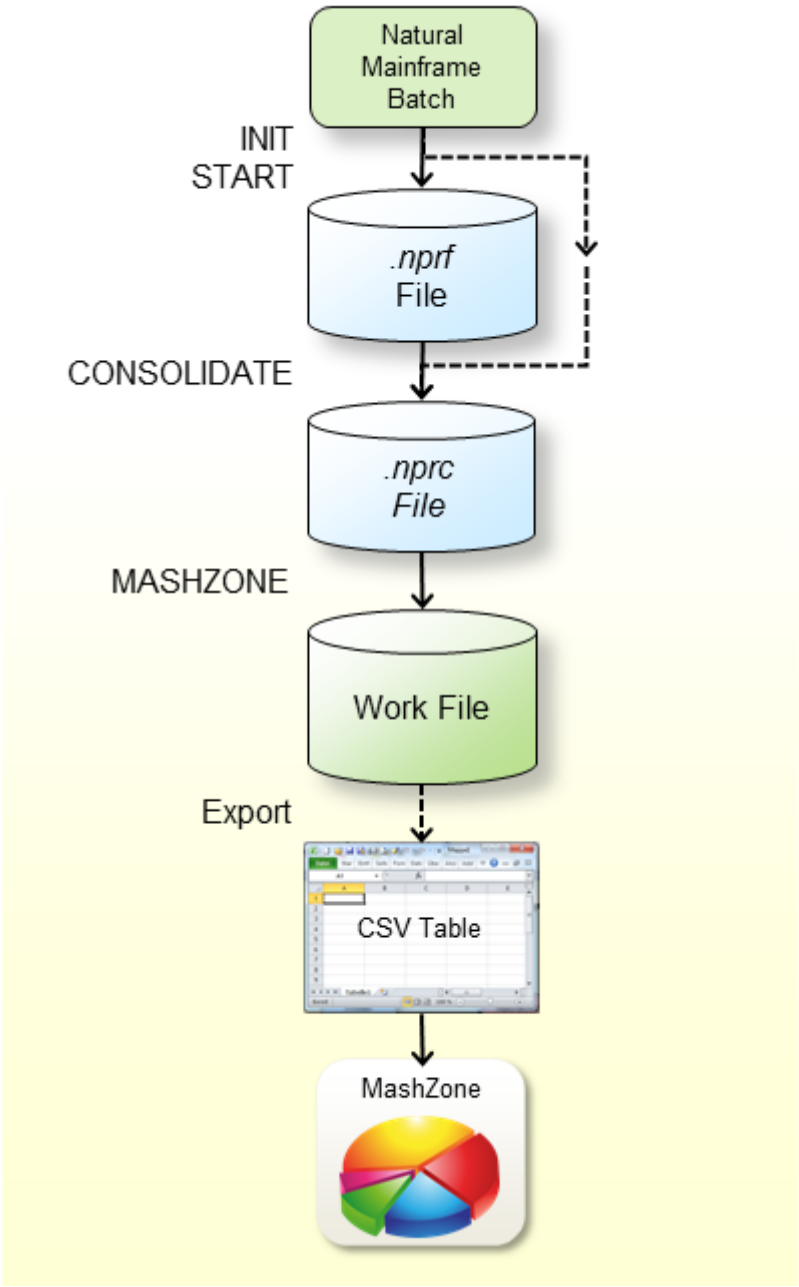
   The MashApps in the `importexport\Profiler_`*`date`* directory are named as follows:

`M_MashAppName version_revision_date-time.mzp`

where *MashAppName* is the name of the MashApp in MashZone which can be either of the following:

| MashAppName | Purpose |
|---|---|
| `Natural Profiler` | Evaluate the Natural Profiler data and the Profiler properties and statistics of a monitored application. |
| `Natural Profiler Compare` | Compare two Natural Profiler data files. |

## Preparing the Profiler Data




The graphic above illustrates the steps you have to perform before you can evaluate the Natural Profiler data in MashZone:

- Profile the Natural mainframe batch application with the Natural Profiler data collection functions as described in the section ***Using the Profiler Utility in Batch Mode***. The Profiler writes the event data to an `.nprf` Natural Profiler resource file.
- Consolidate the event data using the Profiler utility `CONSOLIDATE` function. The consolidated data is written to an `.nprc` Natural Profiler resource consolidated file.
- Alternatively, you can specify `CONSOLIDATE=ON` with the Profiler utility `INIT` function when you profile the Natural mainframe batch application. In this case, the Profiler writes the event data directly to an `.nprc` Natural Profiler resource file.
- Write the consolidated event data with the Profiler utility `MASHZONE` function in CSV (comma-separated values) format to Work File 7.
- Export the data from Work File 7 with any tool (such as FTP) to the Profiler data directory (see ***Unpacking the Zip File***). Use `.csv` as the file extension.
- Enter a reference to the new file in the `Overview.csv` file in the `resources\Profiler` directory.

If you start MashZone, you will find the description of the new file in the **Input** selection box. If you select the line with the description, the Natural Profiler MashApps read the event data from the corresponding CSV file.

If you already started the Natural Profiler MashApp earlier, MashZone may not immediately detect the new entry in the `Overview.csv` file. In this case, start any other MashApp, and then restart the Natural Profiler MashApp to clear the internal MashZone buffer.

## Opening the MashApps

After you have specified all required information as described in the previous sections, you can proceed as follows:

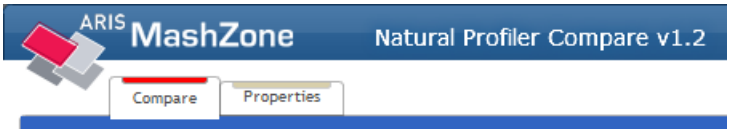
1. Invoke MashZone.
2. Open the Natural Profiler MashApp or the Natural Profiler Compare MashApp.

   The Natural Profiler MashApp offers two tabbed pages for analyzing the Profiler event data and viewing the Profiler properties and statistics:

The **Evaluation page** provides the Profiler event data evaluation.

The **Properties page** lists the Profiler properties and the statistics of the monitored application.

The Natural Profiler Compare MashApp offers two tabbed pages for comparing the Profiler event data and the Profiler properties and statistics:



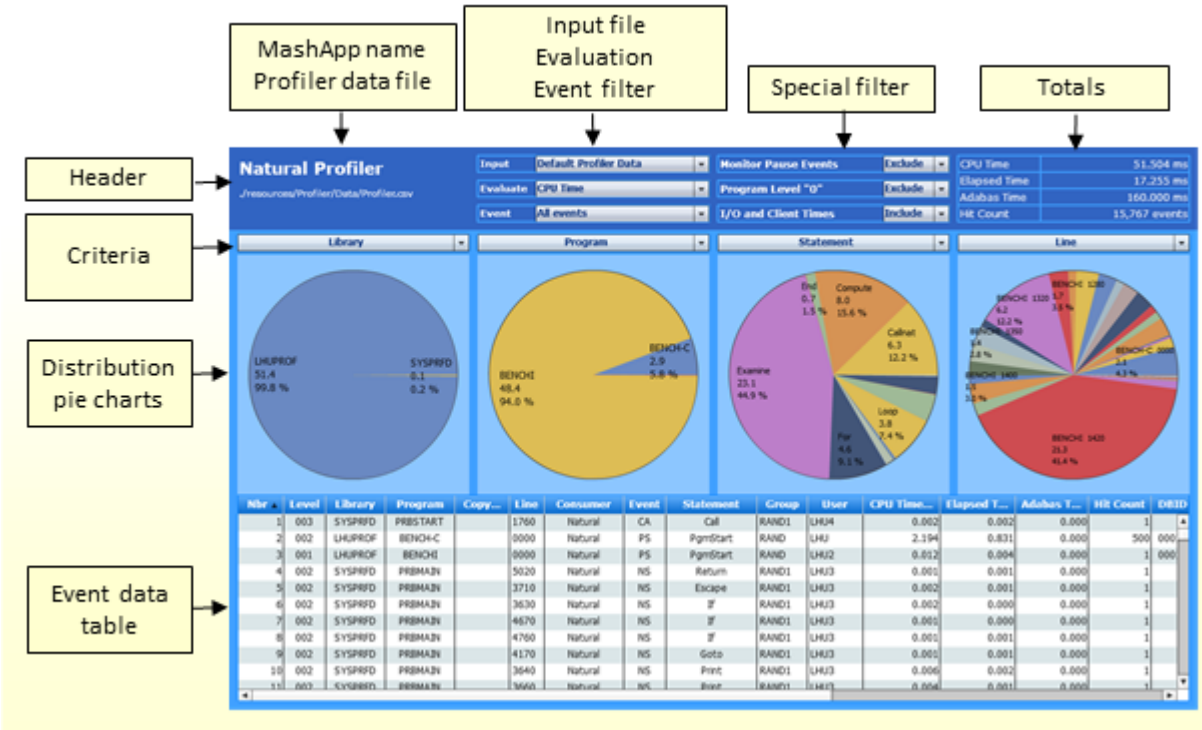

The **Compare page** compares the Profiler event data of two monitored applications.

The **Properties page** lists the Profiler properties and the statistics of the two monitored applications.

The pages are described in the following section.

## Evaluation Page

The **Evaluation** page (Natural Profiler MashApp) looks similar to the example below:




The **Evaluation** page is organized in the following sections:

- The **header** at the top of the page with **Input** and KPI selection fields, filters and totals;
- The selection boxes for the distribution criteria and corresponding **distribution pie charts**;
- The **event data table** at the bottom of the page with the consolidated event data.

This section covers the following topics:

- Evaluation Header
- Distribution Pie Charts
- Event Data Table

## Evaluation Header

The header contains the following elements (from left to right and top to bottom):

- The name of the MashApp.
- The path and name of the Profiler data file currently selected.
- The **Input** selection box which is used to select the Profiler data file. The file names listed for selection are taken from the `Description` column in the `Overview.csv` file. See *Editing the Overview.csv Resource File*. The selected file is used for both pages of the Natural Profiler MashApp.

- The **Evaluate** selection box which is used to select the KPI you want to evaluate in the pie charts. The following KPIs are available:

  CPU Time
  Elapsed Time
  Adabas Command Time
  Hit Count

  The CPU time is evaluated by default. All time values are expressed in milliseconds.
- The **Event** selection box is used to filter the event type you want to evaluate. The event types available for selection depend on the event types collected with the Natural Profiler. The pie charts, the event data table and the totals reflect only the data returned for the selected event types. By default, all event types are evaluated.

  Filtering specific event types is especially useful, for example, to evaluate the hit count of events that seldom occur such as error events.
- The **Monitor Pause Events** selection box is used to filter Monitor Pause events. The filter is valid for the pie charts, the event data table and the totals. By default, the evaluations do not reflect Monitor Pause events. If you include Monitor Pause events, you can see how often monitoring paused, and how long and why it paused.
- The **Program Level "0"** selection box is used to filter events which are executed at Program Level `0`. These events usually relate to the Natural administration rather than the application execution. The filter is valid for the pie charts, the event data table and the totals. By default, the evaluations do not reflect the events at the program level `0`.
- The **I/O and Client Times** selection box is used to filter the I/O time (`IB` event) and the Natural RPC client time (`RW` event). These times mainly measure the user reaction (how long it took to press ENTER), especially when the elapsed time for an interactive application is evaluated. They are less relevant for the application performance. The filter is valid for the pie charts, the event data table and the totals. By default, the evaluations reflect the I/O and client times.
- Summarized totals for the CPU time, the elapsed time, the Adabas time and the hit count according to the values that are currently selected in the header and in the pie charts.

### Distribution Pie Charts

The **Evaluation** page contains four pie charts. Each pie chart shows the distribution of the KPI (selected in the **Evaluate** selection box) for the criterion selected in the box directly above the pie chart (see the example in *Evaluating Distribution Pie Charts*).

This section covers the following topics:

- Criteria for All Event Types
- Criteria for Specific Event Types Only

- Evaluating Distribution Pie Charts

### Criteria for All Event Types

The following criteria are available for all event types:

**Consumer**

The consumer combines one or more event types into a new criterion. The new criterion depends on the process that consumed the CPU or elapsed time given with the event data. For example, the time returned for a Before Database Call (`DB`) event is consumed by the database (and therefore belongs to the **Database** consumer), whereas the time returned for an After Database Call (`DA`) event is consumed by the Natural application (and therefore belongs to the **Natural** consumer).

A consumer evaluation is not relevant for an Adabas time or hit count analysis.

The following consumers are provided:

| Consumer | Event Type | Description |
|---|---|---|
| Administration | `PL`, `PT` | The time Natural used to load and release Natural objects.<br><br>On the mainframe, the loading of Natural objects from the Natural system file is charged to the **Database** consumer (`DB` event against FNAT or FUSER system file).<br><br>On UNIX and Windows, the entire operation is charged to the **Administration** consumer. |
| Database | `DB` | The time consumed for database calls.<br><br>For the CPU time, it is the time spent in the Natural region. |
| External | `CB` | The time spent for external (non-Natural) program calls. |
| I/O | `IB` | The time spent for I/Os.<br><br>When you analyze the elapsed time of an interactive application, this section shows the user response time.<br><br>This section is only displayed if **I/O and Client Times** is included in the selection box in the page header. |
| Pause | `MP` | The time for which the monitor paused.<br><br>This section is only displayed if **Monitor Pause Events** is included in the selection box in the page header. |
| RPC Client | `RW` | The time spent on the Natural RPC client side.<br><br>When you analyze the elapsed time of an interactive RPC application, this section shows the user's response time. |

| Consumer | Event Type | Description |
|---|---|---|
| | | This section is only displayed if **I/O and Client Times** is included in the selection box in the page header. |
| RPC Server | RI, RO | The time consumed by the Natural RPC server layer. |
| Session | SI, ST | The time required to initialize the Natural session. |
| Natural | CA, DA, E, IA, NS, PR, PS, RS, U | The time Natural spent executing the program code. |

**Event**

The type of the event to be evaluated. All event types are listed in *Events and Data Collected* in the section *Using the Profiler Utility in Batch Mode*.

For technical reasons, a Program Resume (PR) event uses the same timestamp as the Program Termination (PT) event that immediately precedes the PR event. Therefore, the PT event generally shows a CPU time and elapsed time of zero (0).

**Group**

The group ID for Natural RPC applications running under Natural Security.

**Level**

The level at which the profiled program executes.

**Library**

The Natural library that contains the profiled program.

**Line**

The source line in which the Natural statement executed by the profiled program is coded.

**Line100**

Source lines with similar line numbers (rounded down to the next multiple of 100).

**Program**

The name of the profiled program.

**Statement**

The Natural statement (for example, EXAMINE) executed in the profiled program.

**User**

The user ID for Natural RPC applications running under Natural Security.

#### Criteria for Specific Event Types Only

The following criteria are only available for specific event types. If you select an event-specific criterion, the pie chart will only reflect the data of the related events.

**Client User**
The Natural RPC client user ID type for `RI`, `RO` and `RW` events.

**Command**
The Adabas command for `DB` and `DA` events.

**File**

The database ID and file number of the Natural system file for `PS` and `PT` events.
The database ID and file number of the Adabas file accessed for `DB` and `DA` events.

**Return Code**

The termination return code for `ST` events.
The database response and subcode for `DA` events.
The subprogram response code for `CA` events.
The error number for `E` events.
The Natural RPC return code for `RI`, `RO` and `RW` events.

**Target Program**

The session backend program name for `ST` events.
The target program name for `PL` events.
The name of the called subprogram for `CB` and `CA` events.
The error handling program name for `E` events.
The Natural RPC subprogram name for `RS` events.

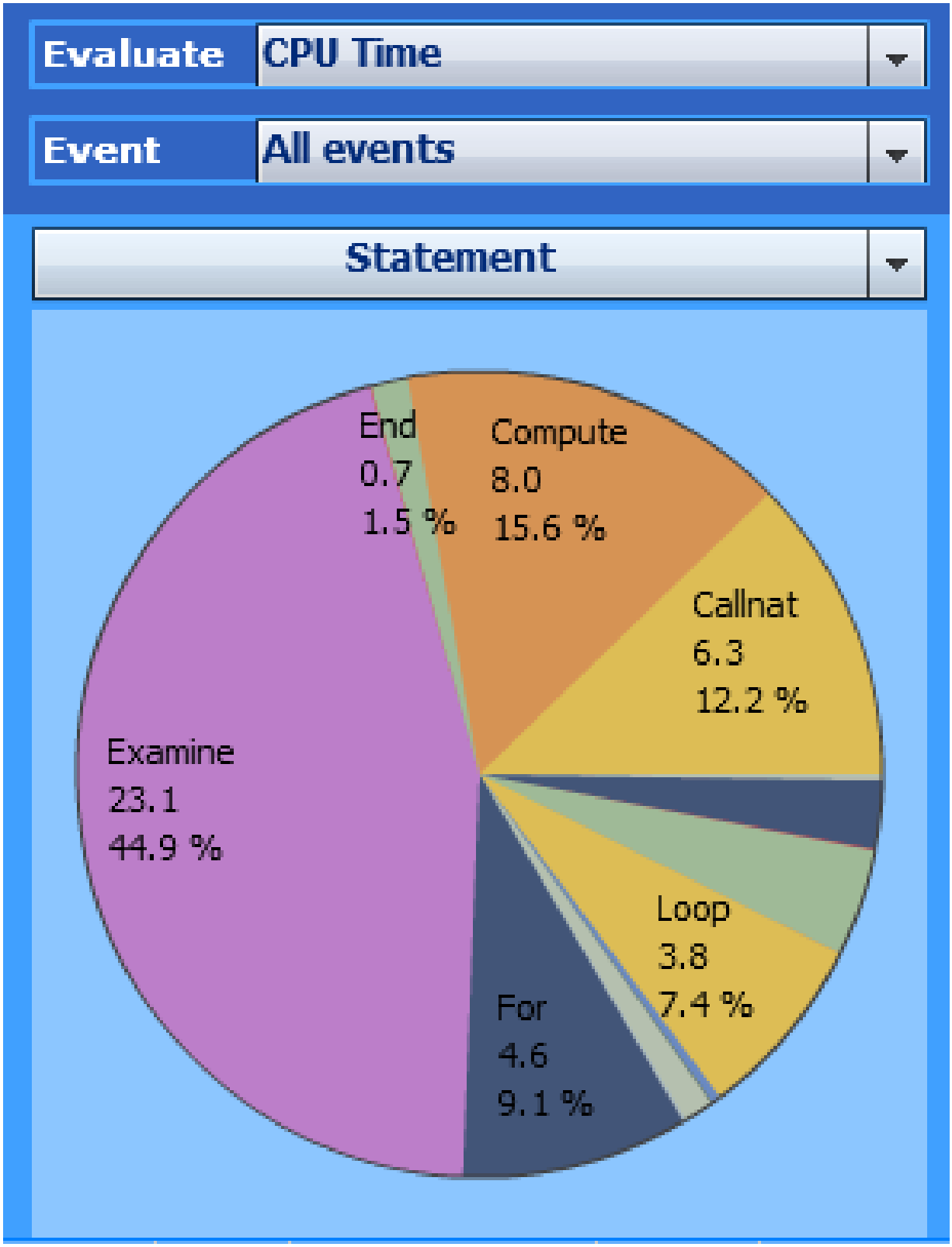
**Type**

The program type for `PS` and `PT` events.
The monitor pause reason for `MP` events.
The user event subtype for `U` events.
The return code indicator (system or user) for `ST` events.

#### Evaluating Distribution Pie Charts

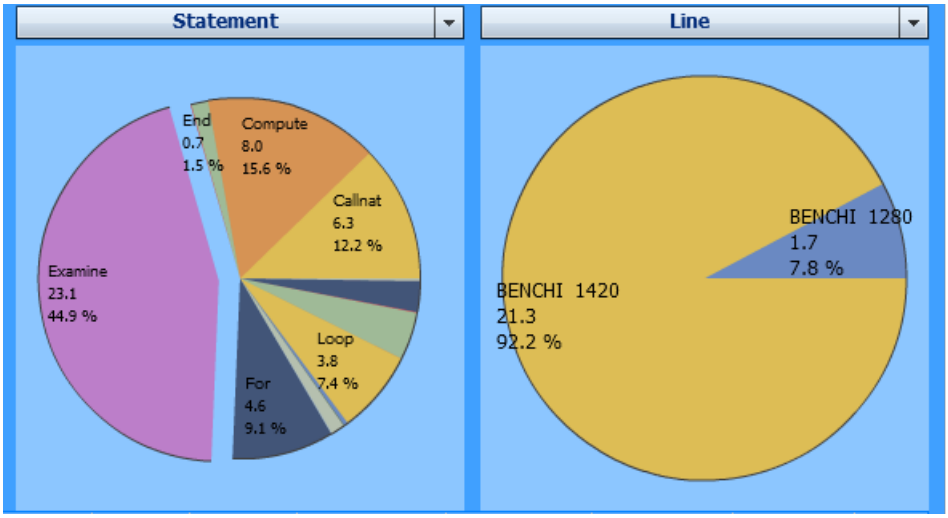
This section describes how you can evaluate distribution pie charts.

- A distribution pie chart shows the distribution for the criterion currently selected in the selection box directly above the pie chart. In the following example, **Statement** has been selected as the criterion for evaluating the CPU time:




The pie chart shows the distribution of the CPU time for the used Natural statements. It indicates that the **Examine** statement consumed the most CPU time (`23.1` ms / `44.9` percent).

- If you click on a segment in the pie chart, all following pie charts, the event data table and the totals use the selected value as the filter criterion. In the example below, the **Examine** statement in the left pie chart has been selected:




The right pie chart above displays only those two lines in which an **Examine** statement is executed. The event data table at the bottom of the page and the totals in the page header also reflect the data for the **Examine** statement only.

- To remove a selection, click on the background of a pie chart.
- If you move the cursor to the upper right corner of a pie chart, a drop down list provides the option to save the pie chart as a picture or to display and save the related data. For the display, a window opens with a table containing the data monitored for the Natural statements:

Data feed table: Pie3

| 123 KPI | T KPI3 | T Tooltip(0) | T Tooltip(2) | T Tooltip(4) | T Tooltip(8) |
|---|---|---|---|---|---|
| 0.0 | Call | Statement | Call | CPU Time | ms |
| 6.3 | Callnat | Statement | Callnat | CPU Time | ms |
| 8.0 | Compute | Statement | Compute | CPU Time | ms |
| 0.7 | End | Statement | End | CPU Time | ms |
| 0.0 | Escape | Statement | Escape | CPU Time | ms |
| 23.1 | Examine | Statement | Examine | CPU Time | ms |
| 4.6 | For | Statement | For | CPU Time | ms |
| 0.0 | Goto | Statement | Goto | CPU Time | ms |
| 0.0 | If | Statement | If | CPU Time | ms |
| 0.6 | Ignore | Statement | Ignore | CPU Time | ms |
| 0.0 | Include | Statement | Include | CPU Time | ms |
| 0.1 | Input | Statement | Input | CPU Time | ms |

Save as CSV... Close

In the example above, the table lists the values of the left pie chart in the previous graphic. The **KPI** column lists the CPU time and the **KPI3** column the corresponding Natural statement.

You can save the table data as a CSV (comma-separated values) formatted file.

## Event Data Table

The event data table at the bottom of the **Evaluation** page lists the consolidated Profiler event data according to the values currently selected in the page header and the pie charts. If you click on the table header of a column, the data is sorted by that column.

In the following example, the event data table is sorted by the CPU time (descending):

| Nbr | Level | Library | Program | Copy... | Line | Consumer | Event | Statement | Group | User | CPU Tim... | Elapsed T... | Adabas T... | Hit Count | DBID |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 96 | 001 | LHUPROF | BENCHI | | 1420 | Natural | NS | Examine | RAND | LHU1 | 21.329 | 3.690 | 0.000 | 500 | |
| 142 | 001 | LHUPROF | BENCHI | | 1320 | Natural | PR | Callnat | RAND | LHU2 | 2.208 | 0.744 | 0.000 | 500 | |
| 2 | 002 | LHUPROF | BENCH-C | | 0000 | Natural | PS | PgmStart | RAND | LHU | 2.194 | 0.831 | 0.000 | 500 | 000 |
| 138 | 001 | LHUPROF | BENCHI | | 1320 | Natural | NS | Callnat | RAND | LHU2 | 2.070 | 0.564 | 0.000 | 499 | |
| 141 | 001 | LHUPROF | BENCHI | | 1320 | Administration | PL | Callnat | RAND | LHU2 | 1.929 | 0.563 | 0.000 | 500 | |
| 95 | 001 | LHUPROF | BENCHI | | 1280 | Natural | NS | Examine | RAND | LHU1 | 1.797 | 0.512 | 0.000 | 500 | |
| 140 | 001 | LHUPROF | BENCHI | | 1360 | Natural | NS | Resize | RAND | LHU2 | 1.398 | 0.443 | 0.000 | 500 | |
| 133 | 001 | LHUPROF | BENCHI | | 1230 | Natural | NS | Compute | RAND | LHU2 | 1.280 | 0.434 | 0.000 | 500 | |
| 129 | 001 | LHUPROF | BENCHI | | 1370 | Natural | NS | Compute | RAND | LHU2 | 1.179 | 0.425 | 0.000 | 500 | |
| 134 | 001 | LHUPROF | BENCHI | | 1410 | Natural | NS | Compute | RAND | LHU2 | 1.014 | 0.388 | 0.000 | 500 | |
| 71 | 001 | LHUPROF | BENCHI | | 1310 | Natural | NS | For | RAND | LHU | 0.795 | 0.352 | 0.000 | 500 | |

# Compare Page

The **Compare** page (Natural Profiler Compare MashApp) compares the Profiler event data of two monitored applications as shown in the following example:

The **Compare** page is organized in the following sections:

- The header at the top of the page with **Input** and KPI selection fields, filters and totals.
- The column chart comparing the values of the two monitored applications:

  Values for the first application are shown in the left (green) column, values for the second application are shown in the right (yellow) column.

This section covers the following topics:

- Compare Header

- Compare Column Chart

### Compare Header

The **Compare** header contains the following elements (from left to right and top to bottom):

- The name of the MashApp.
- The paths and names of the two Profiler data files to be compared.
- The **Input 1** and **Input 2** selection boxes with the Profiler data files selected for comparison. The file names listed for selection are taken from the `Description` column in the `Overview.csv` file (see *Editing the Overview.csv Resource File*). The selected files are used for both pages of the Natural Profiler Compare MashApp.
- Summarized totals for the CPU time, the elapsed time, the Adabas time and the hit count according to the values for both applications listed in the header and column chart.
- The **Evaluate** selection box with the KPI to evaluate in the column chart (see *Evaluation Header*).
- The **Event** selection box with the event type to evaluate for (see *Evaluation Header*).
- The **Criterion** selection box with the filter criterion to use for the KPI distribution. The criteria available for selection correspond to the criteria for the distribution pie charts on the **Evaluation** page (see *Evaluating Distribution Pie Charts*.
- The **Monitor Pause Events** selection box with the filter criterion to use for Monitor Pause events (see *Evaluation Header*).
- The **Program Level "0"** selection box with the filter criterion to use for events at the program level `0` (see *Evaluation Header*).
- The **I/O and Client Times** selection box with the filter criterion to use for I/O time (`IB` event) and Natural RPC client time (`RW` event) events; see *Evaluation Header*.
- The **Pre-Selection** values with restrictions for the column chart values to a specific criterion instance, for example to a specific library.

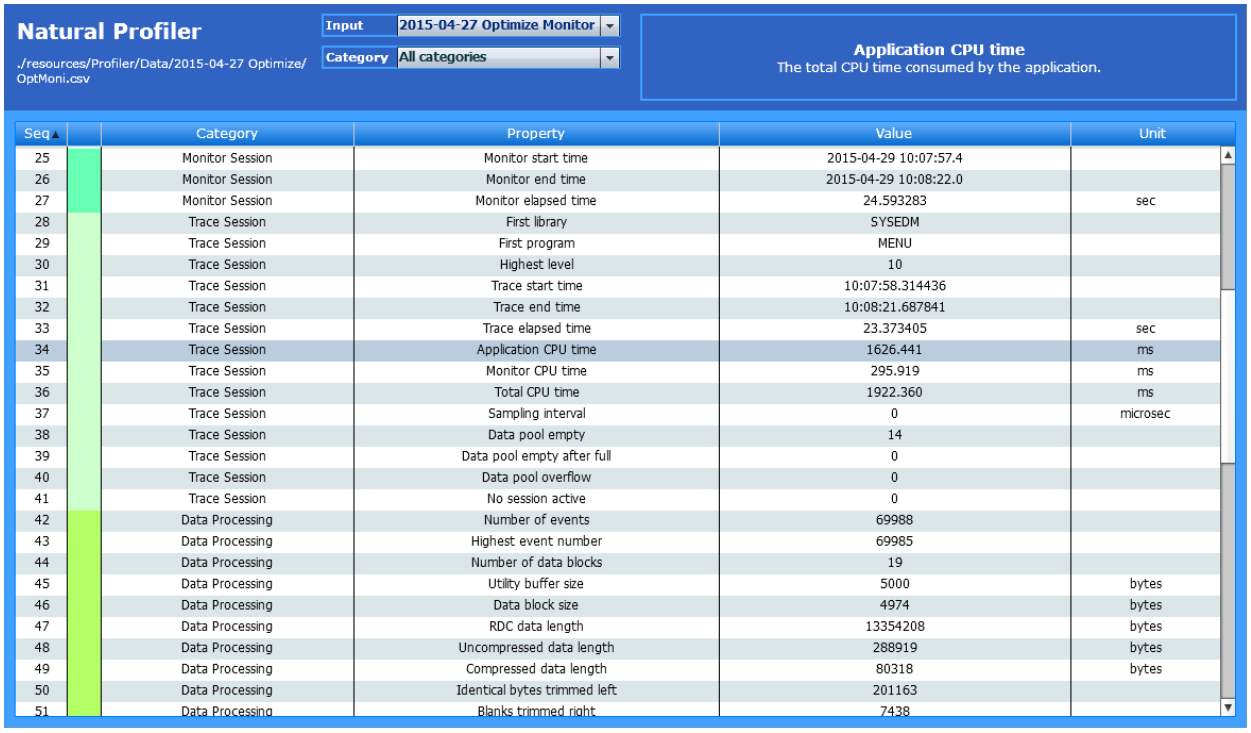
### Compare Column Chart

The **Compare** column chart compares the values of the KPI (selected in the **Evaluate** selection box) for the criterion specified in the **Criterion** selection box for both profiled applications.

In the example of a **Compare page** shown earlier, the CPU time (**Evaluate** selection box) of each program (**Criterion** selection box) executed by `Numeric Operations MF` (**Input 1**) is compared with the corresponding time of `Numeric Operations LUW` (**Input 2**). Additionally, a pre-selection has been specified so that only values from the library `PRFDEMO` are considered. The green columns show the CPU times of `Numeric Operations MF`, the yellow columns the CPU times of `Numeric Operations LUW`.

# Properties Page

The **Properties** page lists the Profiler properties and the statistics of the monitored application as shown in the following example:




| Seq | | Category | Property | Value | Unit |
|---|---|---|---|---|---|
| 25 | | Monitor Session | Monitor start time | 2015-04-29 10:07:57.4 | |
| 26 | | Monitor Session | Monitor end time | 2015-04-29 10:08:22.0 | |
| 27 | | Monitor Session | Monitor elapsed time | 24.593283 | sec |
| 28 | | Trace Session | First library | SYSEDM | |
| 29 | | Trace Session | First program | MENU | |
| 30 | | Trace Session | Highest level | 10 | |
| 31 | | Trace Session | Trace start time | 10:07:58.314436 | |
| 32 | | Trace Session | Trace end time | 10:08:21.687841 | |
| 33 | | Trace Session | Trace elapsed time | 23.373405 | sec |
| 34 | | Trace Session | Application CPU time | 1626.441 | ms |
| 35 | | Trace Session | Monitor CPU time | 295.919 | ms |
| 36 | | Trace Session | Total CPU time | 1922.360 | ms |
| 37 | | Trace Session | Sampling interval | 0 | microsec |
| 38 | | Trace Session | Data pool empty | 14 | |
| 39 | | Trace Session | Data pool empty after full | 0 | |
| 40 | | Trace Session | Data pool overflow | 0 | |
| 41 | | Trace Session | No session active | 0 | |
| 42 | | Data Processing | Number of events | 69988 | |
| 43 | | Data Processing | Highest event number | 69985 | |
| 44 | | Data Processing | Number of data blocks | 19 | |
| 45 | | Data Processing | Utility buffer size | 5000 | bytes |
| 46 | | Data Processing | Data block size | 4974 | bytes |
| 47 | | Data Processing | RDC data length | 13354208 | bytes |
| 48 | | Data Processing | Uncompressed data length | 288919 | bytes |
| 49 | | Data Processing | Compressed data length | 80318 | bytes |
| 50 | | Data Processing | Identical bytes trimmed left | 201163 | |
| 51 | | Data Processing | Blanks trimmed right | 7438 | |

The **Properties** page of the Natural Profiler Compare MashApp lists the properties and statistics of both Profiler data files selected for comparison.

The **Properties** page is organized in the following sections:

- The properties header at the top of the page with **Input** and **Category** selection fields and the property description;
- The properties table with the properties and statistics.

This section covers the following topics:

- Properties Header

- Properties Table

## Properties Header

The header contains the following elements (from left to right and top to bottom):

- The name of the MashApp.
- The path and name of the Profiler data file currently selected.
- The **Input** selection box is used to select the Profiler data file. The names listed for selection are taken from the `Description` column in the `Overview.csv` file. See *Editing the Overview.csv Resource File*. The selected file is used for both pages of the Natural Profiler MashApp.
- The **Category** selection box is used to select a category (listed alphabetically). The selection box only offers the categories for which at least one associated property is found in the Profiler data file.

  If you select a category, the table shows the properties of the selected category only. By default, all categories are displayed. The following categories are available:

| Category | Description |
|---|---|
| Data Consolidation | Statistics of the data consolidation such as the consolidation factor |
| Data Processing | Statistics of the data processing, data compression and data transfer such as the number of events and the compression rate |
| Event Type Statistics | Statistics of the event types such as the number of Program Load events |
| General Info | Information related to the environment and the Natural Profiler such as the internal Profiler version |
| Monitor Pause Statistics | Statistics of Monitor Pause events such as the number of Profiler data pool full situations |
| Monitor Session | Statistics of the Profiler monitor session such as the monitor elapsed time |
| Profiler Resource File | Information related to the Profiler resource file such as the resource name and library |
| Trace Session | Statistics of the Profiler trace session including the application execution such as the CPU time of the total session |

- The **Property** description. If you click on a line in the properties table, the name of the corresponding property and a detailed description of it are displayed in the page header.

### Properties Table

The properties table lists all collected Profiler properties and application statistics. If you click on an entry in the table header of a column, the entire table is sorted by this column. Each color in the second column corresponds to one category.

All Profiler categories and properties are described in detail in the section ***Profiler Statistics***.

## Use Cases

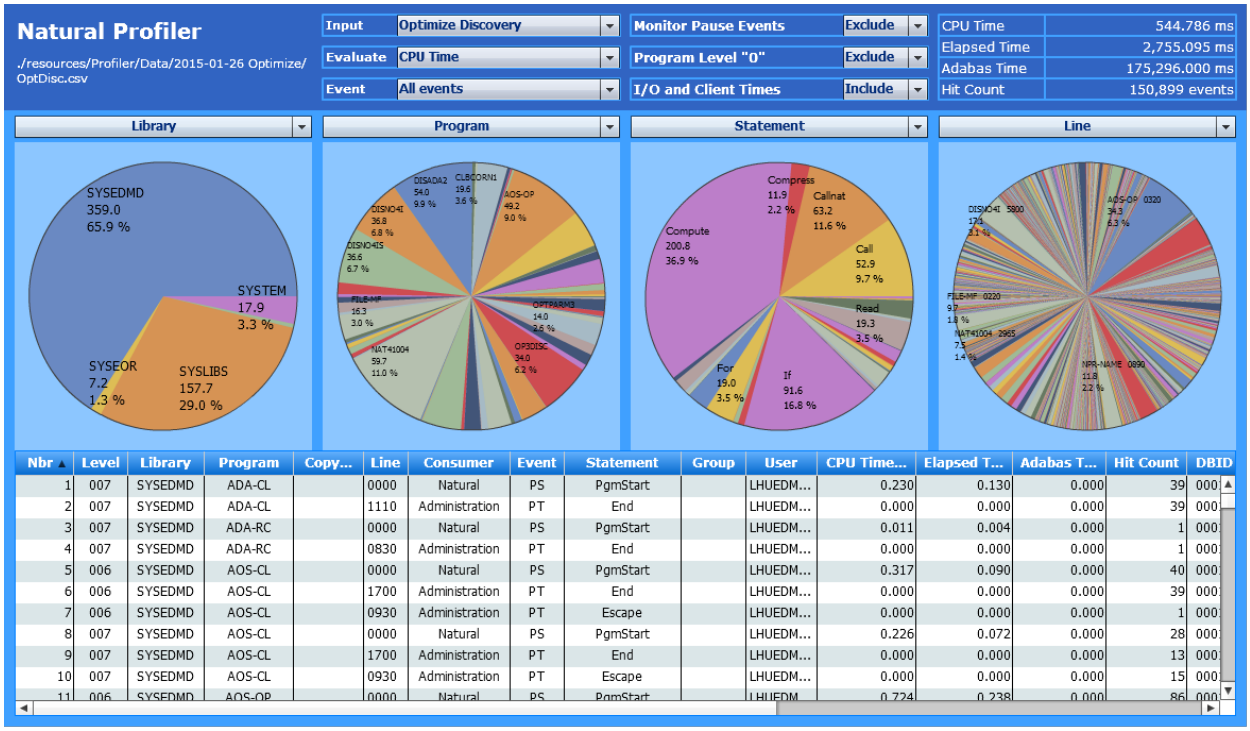
This section describes the following use cases:

- Application Performance Analysis
- Combined Line Numbers
- Consumer
- Natural RPC Server Evaluation
- Natural RPC Server Statistics
- Adabas Command Time Analysis
- Adabas Statistics
- Application Statistics

### Application Performance Analysis

By default, the Natural Profiler MashApp is set up to create CPU time performance analyses of libraries, programs, statements and source lines.

Each pie chart in the example below shows the distribution of the CPU time for each criterion selected:




You can immediately see which library, program, statement or line has consumed how much of the CPU time.

The example above uses the following selections:

**Evaluate:** CPU Time

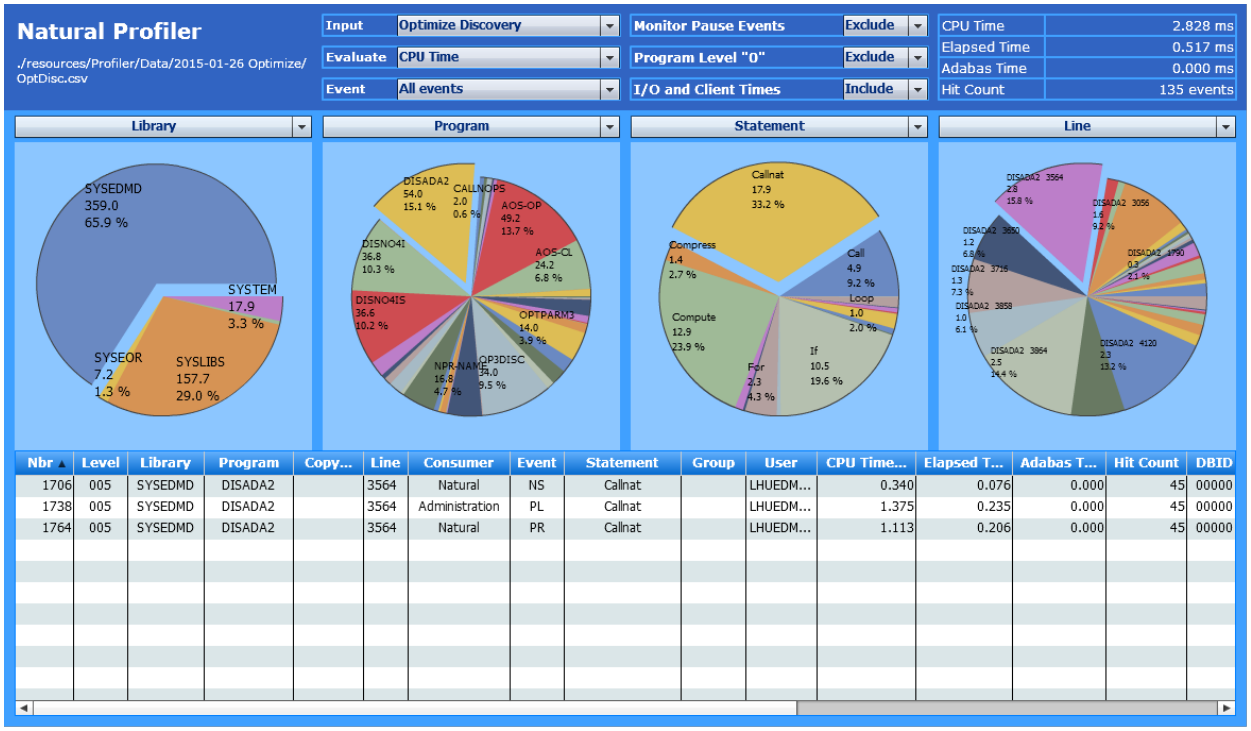
**Event:** All events

**Criteria:** Library, Program, Statement, Line

A large application, such as the example above, references many program lines, thus making it difficult to analyze the corresponding pie chart.

If you click on a segment in a pie chart, the corresponding value of that segment is used as a filter and the amount of data which is displayed in the following pie charts is reduced accordingly. In the example above, a click on the segment with the `SYSEDMD` library in the leftmost pie chart would change the contents of the other three pie charts and only show the programs, statements and lines executed in the `SYSEDMD` library.

The following example refers to the previous one and assumes that in addition to the `SYSEDMD` library, the program `DISADA2`, the `Callnat` statement and the line `3564` are selected in the rightmost chart:

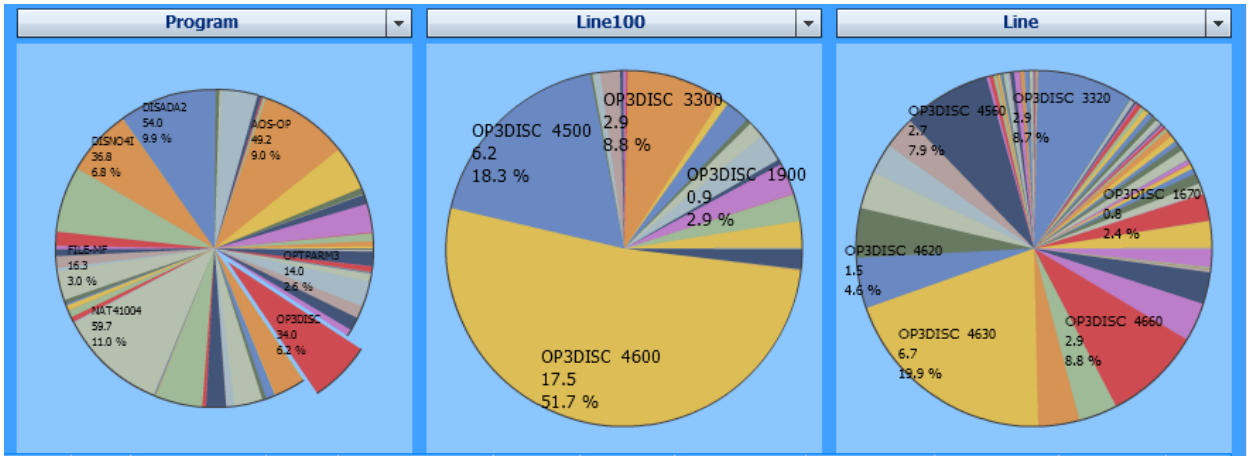



The event data table and the total in the page header now only refer to Line 3564 where the program executes the `CALLNAT` statement. The `CALLNAT` statement caused the event types (`NS`, `PL` and `PR`), each executing `45` times which results in a total **Hit Count** of `135` events.

## Combined Line Numbers

The **Line100** criterion is another approach to reduce the number of entries in the line number chart. It replaces the lines by the previous multiples of 100, thus, combining lines with similar line numbers in one segment of the pie chart.

The example below assumes that you want to find out which part of the program `OP3DISC` consumed the most CPU time. Therefore, you select `OP3DISC` in the **Program** chart so that all other charts only display the data for this program:




The **Line** chart clearly indicates that the statement in the segment of line `4630` uses `19.9` percent of the program's CPU time. However, all other segments are rather small and it is difficult to tell them apart.

The **Line100** chart shows that more than half of the time was consumed by the statements in the lines ranging from 4600 through 4690. Additionally, considering the statements in the lines ranging from 4500 through 4590, this part of the program even consumes 70 percent of the entire execution time. Thus, this program is most busy with the statements in these lines.

The example above uses the following selections:

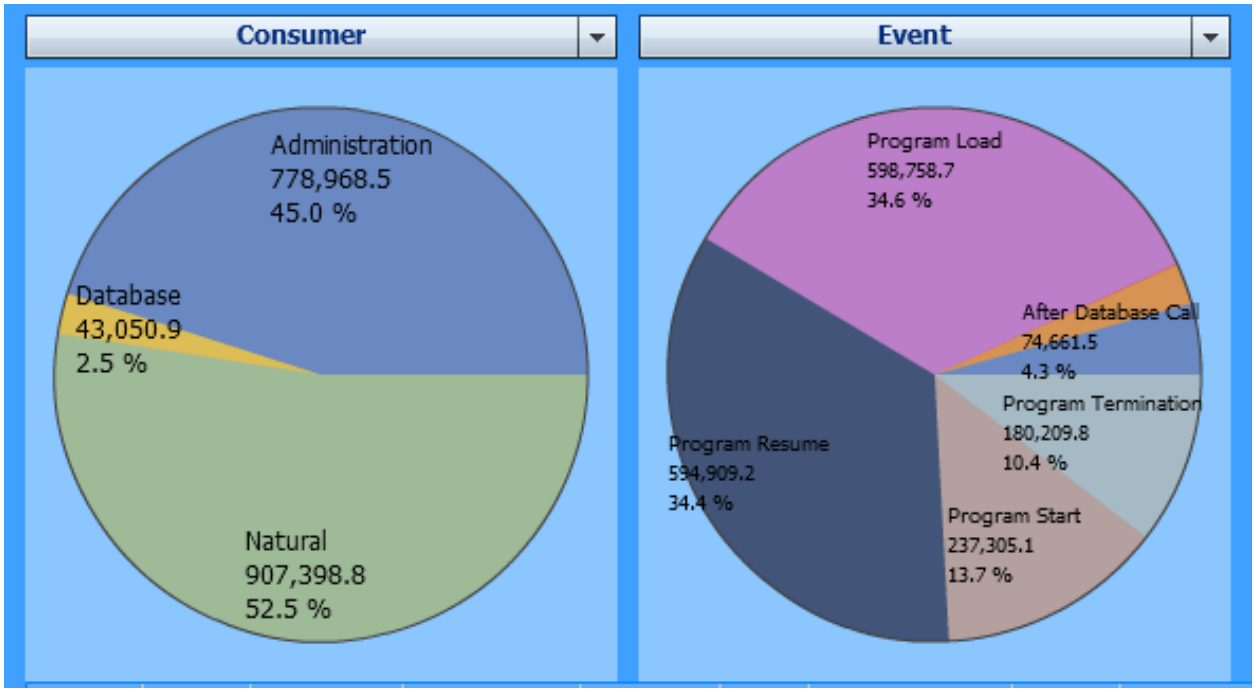
**Evaluate:** CPU Time

**Event:** All events

**Criteria:** Program, Line, Line100

### Consumer

The **Consumer** analysis gives a quick overview of the processes that consumed the most CPU time such as external programs, database calls, I/Os, administration tasks or program instructions. For example:




In the example above (Natural for UNIX, without statement events), 45 % of the CPU time was consumed by administration tasks. A potential reason for this can be the usage of small subprograms which solely call other tiny subprograms. This keeps Natural busy with administration tasks (program load with buffer pool management and program termination), while the time used for executing the code itself is relatively short.

The example above uses the following selections:

**Evaluate:** CPU Time

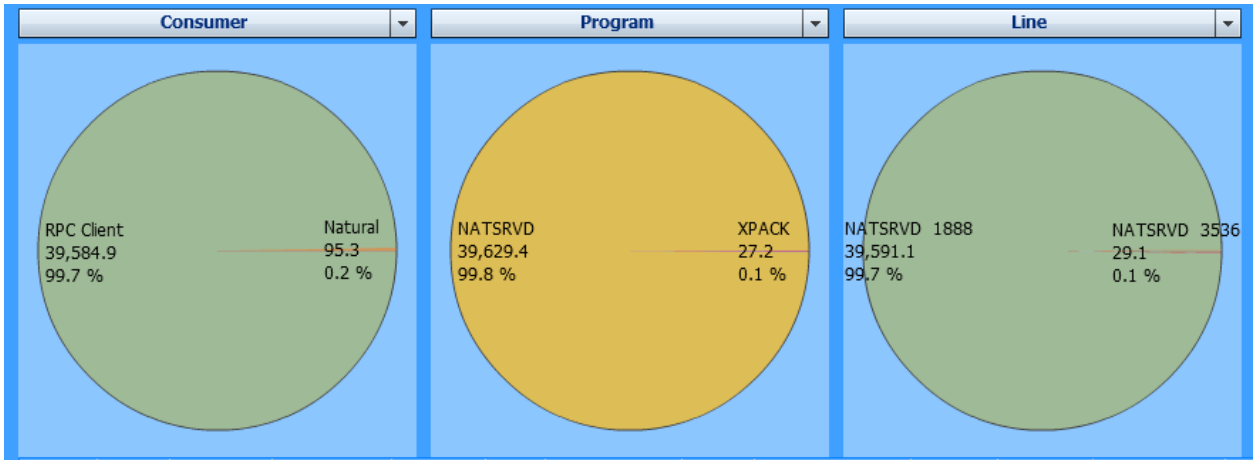
**Event:** All events

**Criteria:** Consumer, Event

### Natural RPC Server Evaluation

When analyzing the elapsed time of an interactive application, waiting for a user response usually takes the most time. For a Natural RPC application, this time is monitored with the RPC Wait for Client (RW) event or the RPC Client consumer.

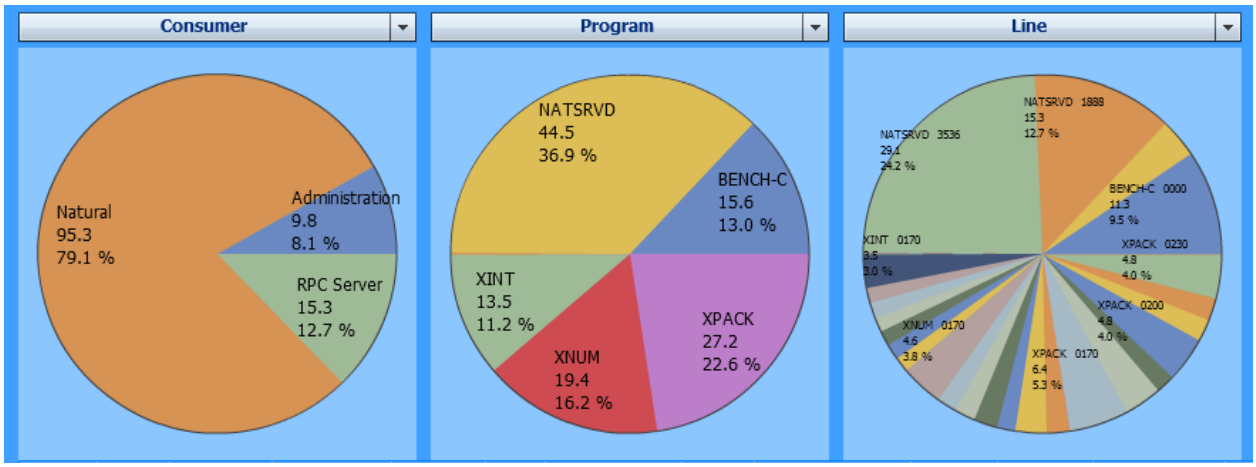
In the example below, the Natural RPC client consumes nearly all of the elapsed time:




The example above uses the following selections:

| | |
|---|---|
| **Evaluate:** | Elapsed Time |
| **Event:** | All events |
| **I/O and Client Times:** | Include |
| **Criteria:** | Consumer, Program, Line |

The MashApp offers a selection field to exclude the client time. If you exclude **I/O and Client Times**, all individual processes performed in the server application are shown similar to the example below:




The example above uses the following selections:

| | |
|---|---|
| **Evaluate:** | Elapsed Time |
| **Event:** | All events |
| **I/O and Client Times:** | Exclude |

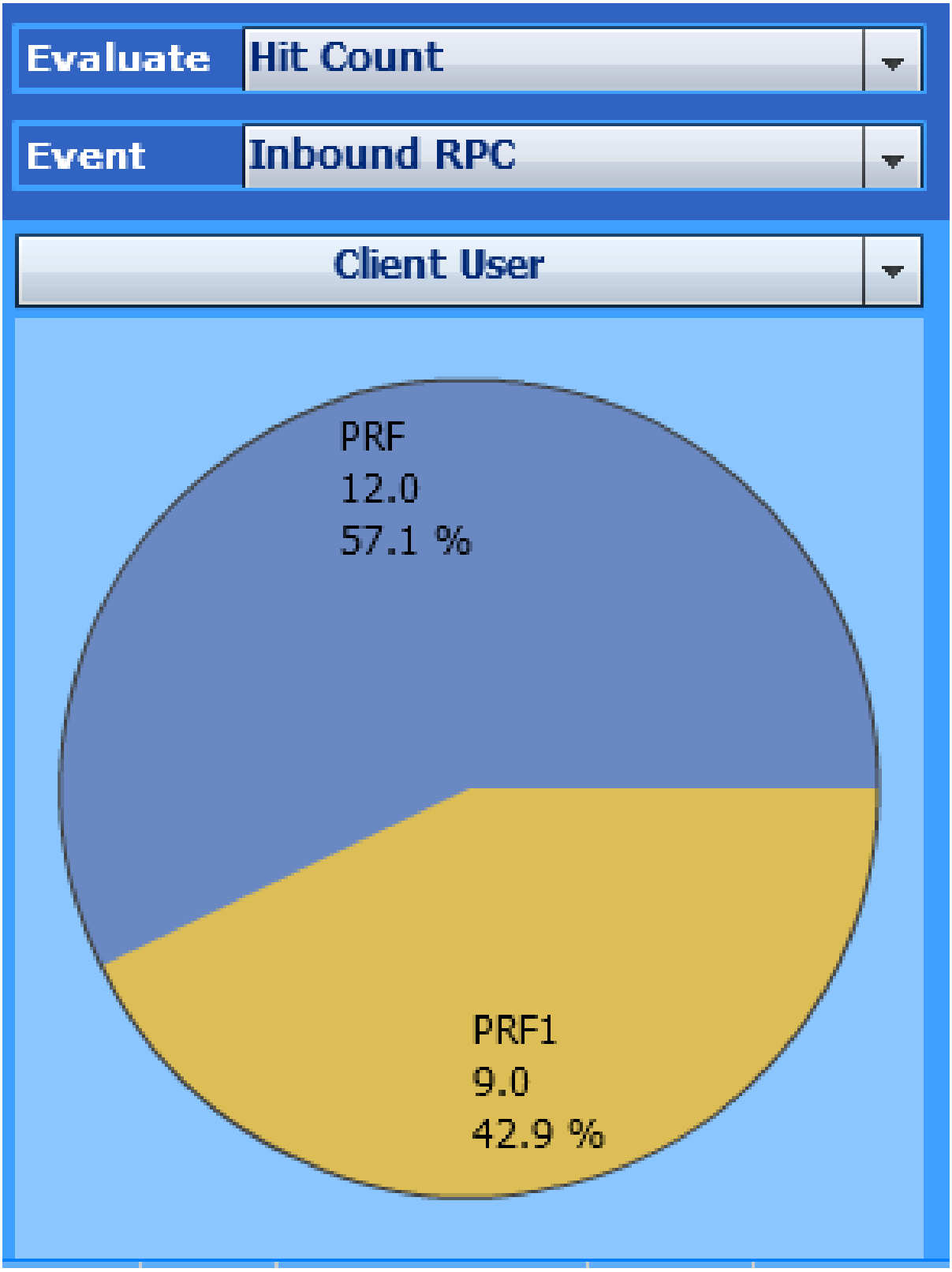
**Criteria:** Consumer, Program, Line

### Natural RPC Server Statistics

You can obtain statistics on remote procedure calls by evaluating the hit count.

**Example of a Natural RPC Client User Evaluation**

The following example shows which user issued Natural RPC requests and how often:




In the example above, the user `PRF` issued `12` Natural RPC requests.

The example uses the following selections:

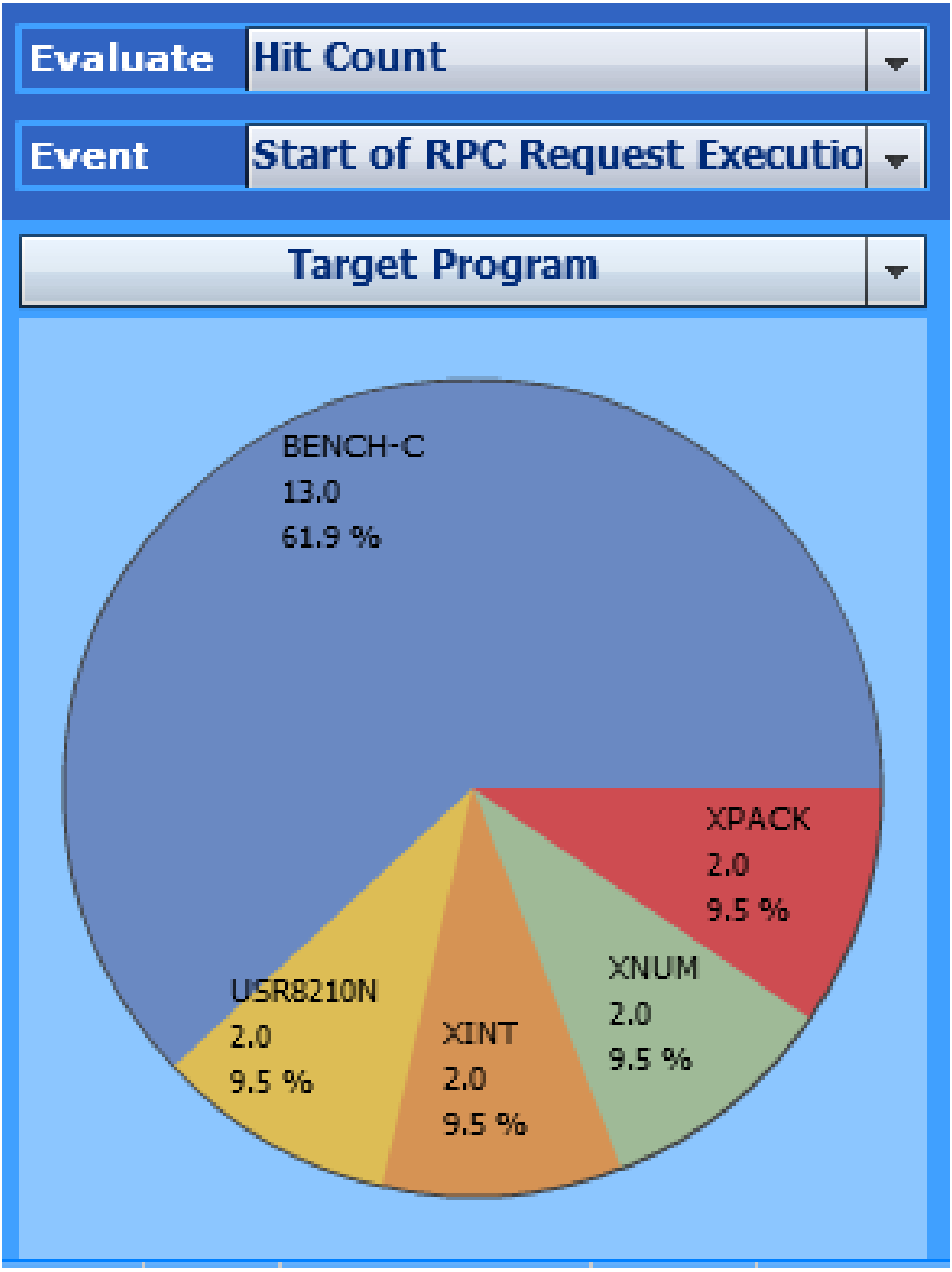
**Evaluate:** Hit Count

**Event:** Inbound RPC

**Criterion:** Client User

**Example of a Natural RPC Target Program Evaluation**

The following example displays which target program was called on the server and how often:




In the example above, `13` Natural RPC requests were issued for the server program `BENCH-C`.

The example uses the following selections:

**Evaluate:** Hit Count

**Event:** Start of RPC Request Execution

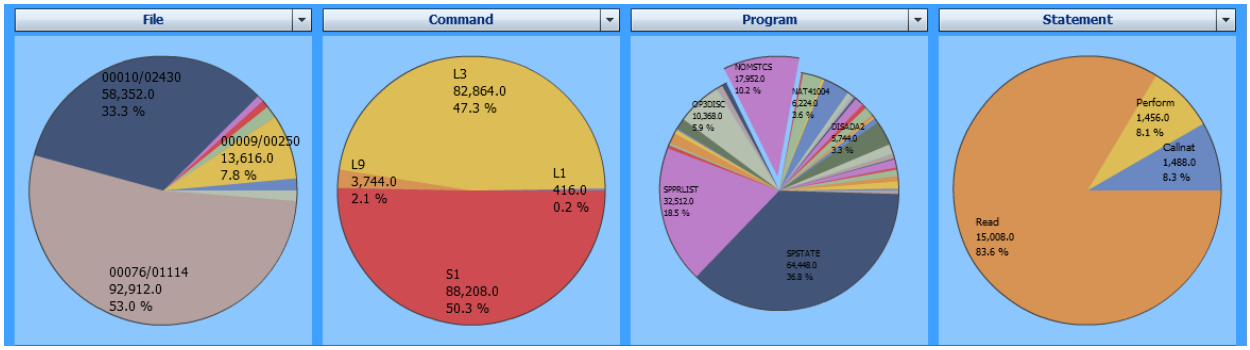
**Criterion:** Target Program

In both examples shown above, single event types are used for the event selection so not to mix the data with other events. For example, if **All events** is selected, the target programs of external program calls (`CA` events) are also displayed in the chart.

### Adabas Command Time Analysis

When the Natural application issues an Adabas command, the database returns the elapsed time the Adabas nucleus required to process the command.

The example below analyzes the distribution of the Adabas command time for the accessed files and for the used Adabas commands. The chart also shows the programs and Natural statements that consumed the Adabas command time.




The most Adabas command time was consumed by calls against the file `1114` of the database `76` and by the Adabas commands `S1` (find record) and `L3` (read logical sequential record).

If you could click on a segment in the pie chart below **File**, you would see the commands issued against the selected file and how much time they consumed. Since the segment of the program `NOMSTCS` is selected in the third pie chart, the fourth pie chart only shows the Adabas command time used by the statements in `NOMSTCS`.

The example uses the following selections:

**Evaluate:** Adabas Command Time

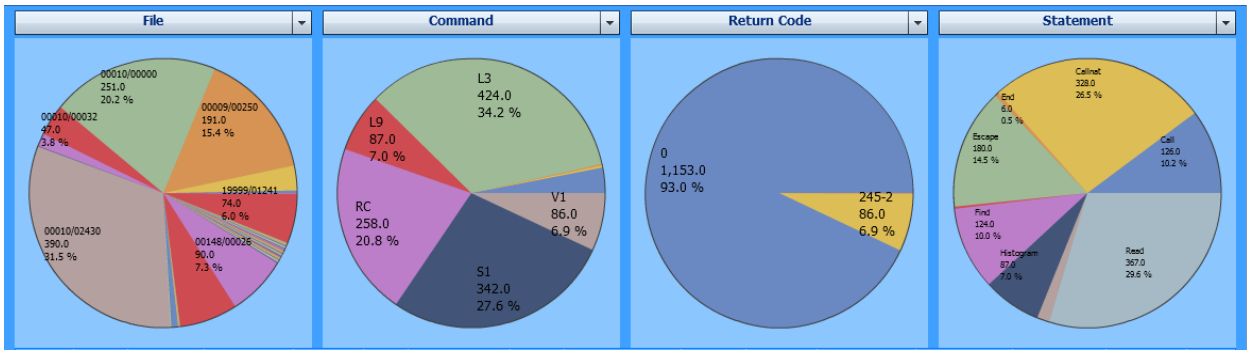
**Event:** All events

**Criteria:** File, Command, Program, Statement

### Adabas Statistics

You can obtain statistics on Adabas requests by evaluating the hit count.

The following example shows which files have been accessed, which commands have been issued, which Adabas response codes have occurred and which Natural statements have issued Adabas requests and how often:




The most Adabas requests were issued against the file `2430` of database `10` and the most frequently Adabas command used was `L3` (read logical sequential record). Most calls were successful (Adabas response 0) but `86` calls received an Adabas response `245` with subcode `2`. The fourth pie chart shows that `READ` statements issued `367` Adabas calls.

The example uses the following selections:

| | |
|---|---|
| **Evaluate:** | Hit Count |
| **Event:** | After Database Call |
| **Criteria:** | File, Command, Return Code, Statement |

### Application Statistics

The following Profiler MashApp examples may answer common statistics questions about monitored Natural applications.

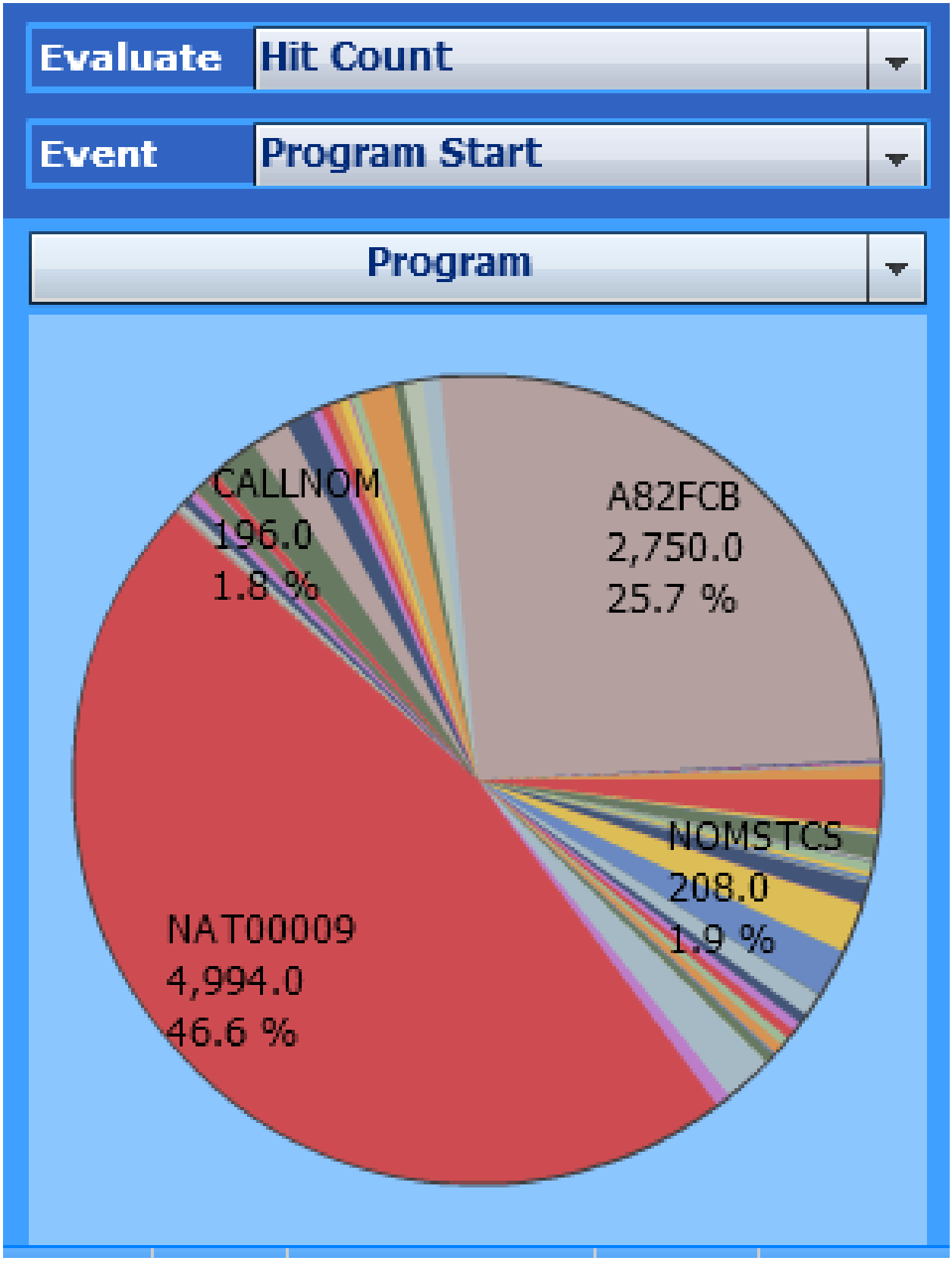
- How often were Natural objects started?
- How many statements were executed in the monitored programs?
- Which objects were called by a selected program and how often?
- How often was a selected object called?

- How many runtime errors occurred?

**How often were Natural objects started?**

Use the following selections to find out:

**Evaluate:** Hit Count
**Event:** Program Start
**Criterion:** Program

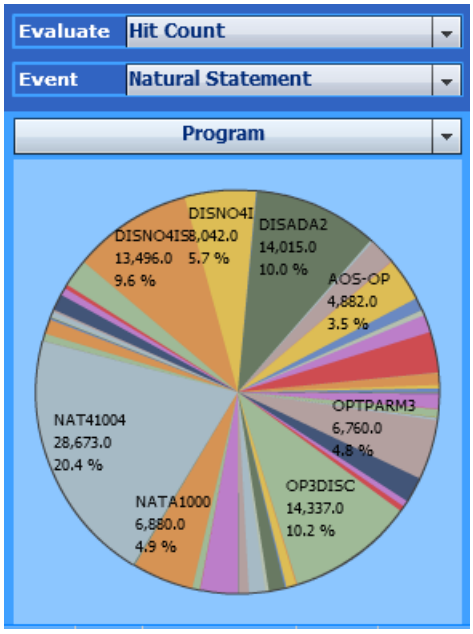

Evaluate
Hit Count
Event
Program Start
Program
CALLNOM
196.0
1.8 %
A82FCB
2,750.0
25.7 %
NOMSTCS
208.0
1.9 %
NAT00009
4,994.0
46.6 %

In the example above, the program `NAT00009` started `4,994` times.

#### How many statements were executed in the monitored programs?

Use the following selections to find out:

**Evaluate:** Hit Count
**Event:** Natural Statement
**Criterion:** Program

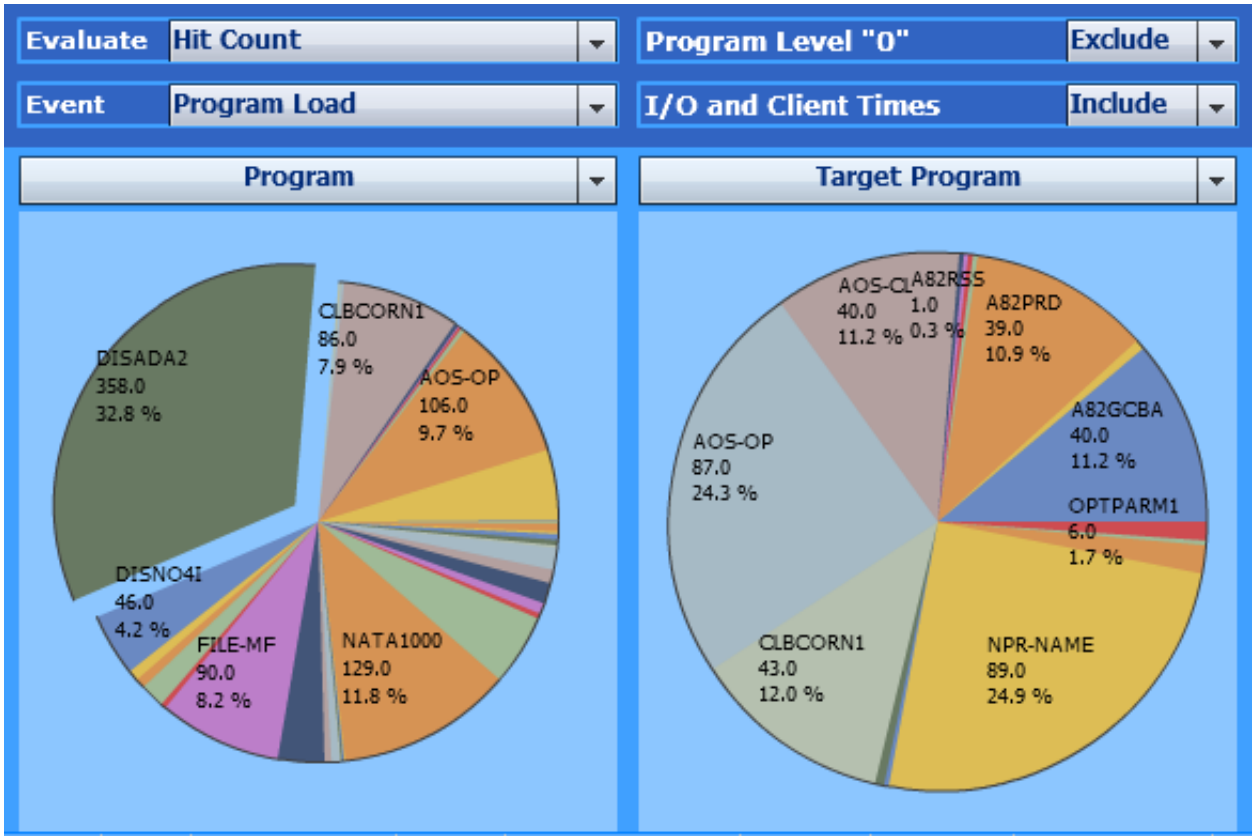



In the example above, the program `NAT41004` executed `28,673` statement events.

#### Which objects were called by a selected program and how often?

Use the following selections to find out:

**Evaluate:** Hit Count
**Event:** Program Load
**Criteria:** Program, Target Program

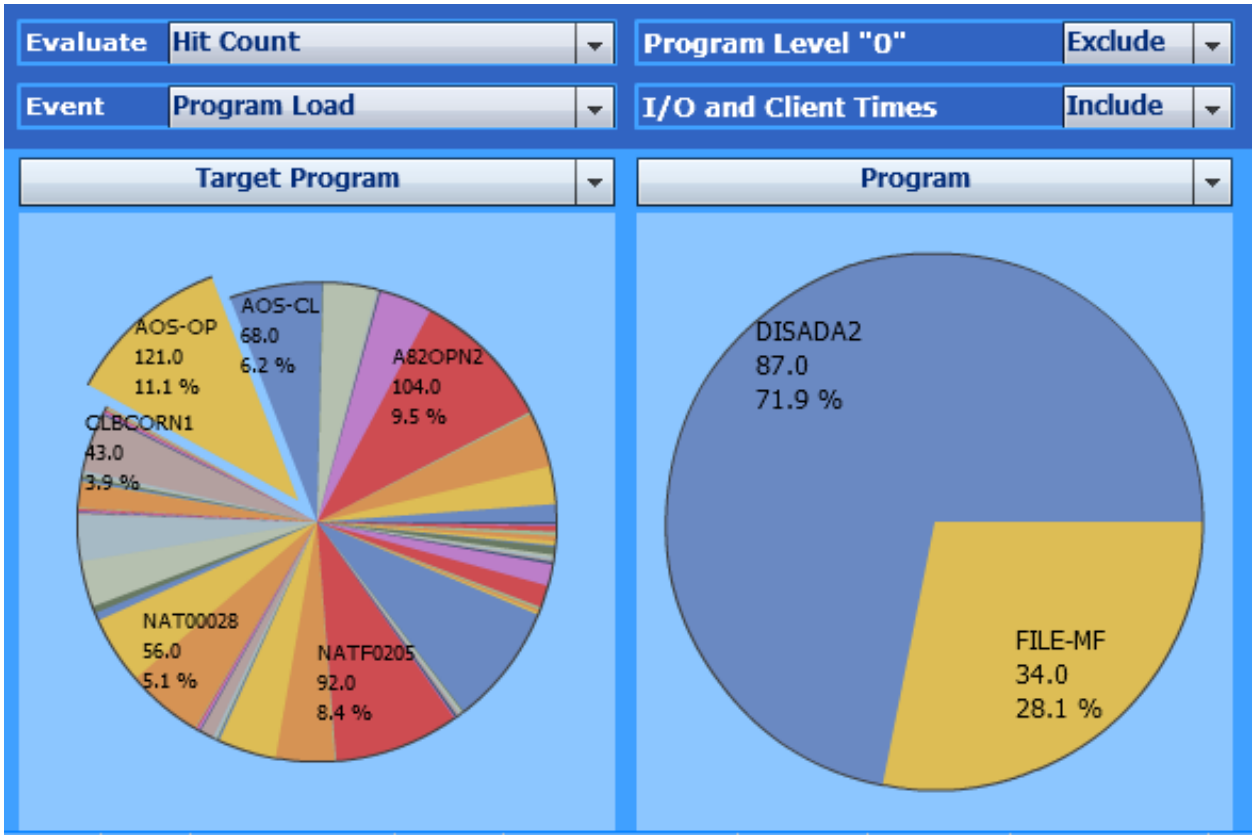



In the example above, the right pie chart shows the Natural objects called by `DISADA2`. The Natural object `NPR-NAME` was called `89` times.

#### How often was a selected object called?

Use the following selections to find out:

**Evaluate:** Hit Count
**Event:** Program Load
**Criteria:** Target Program, Program




In the example above, the right pie chart shows the objects which called the program `AOS-OP`. `AOS-OP` was called `87` times by the program `DISADA2` and `34` times by the program `FILE-MF`.

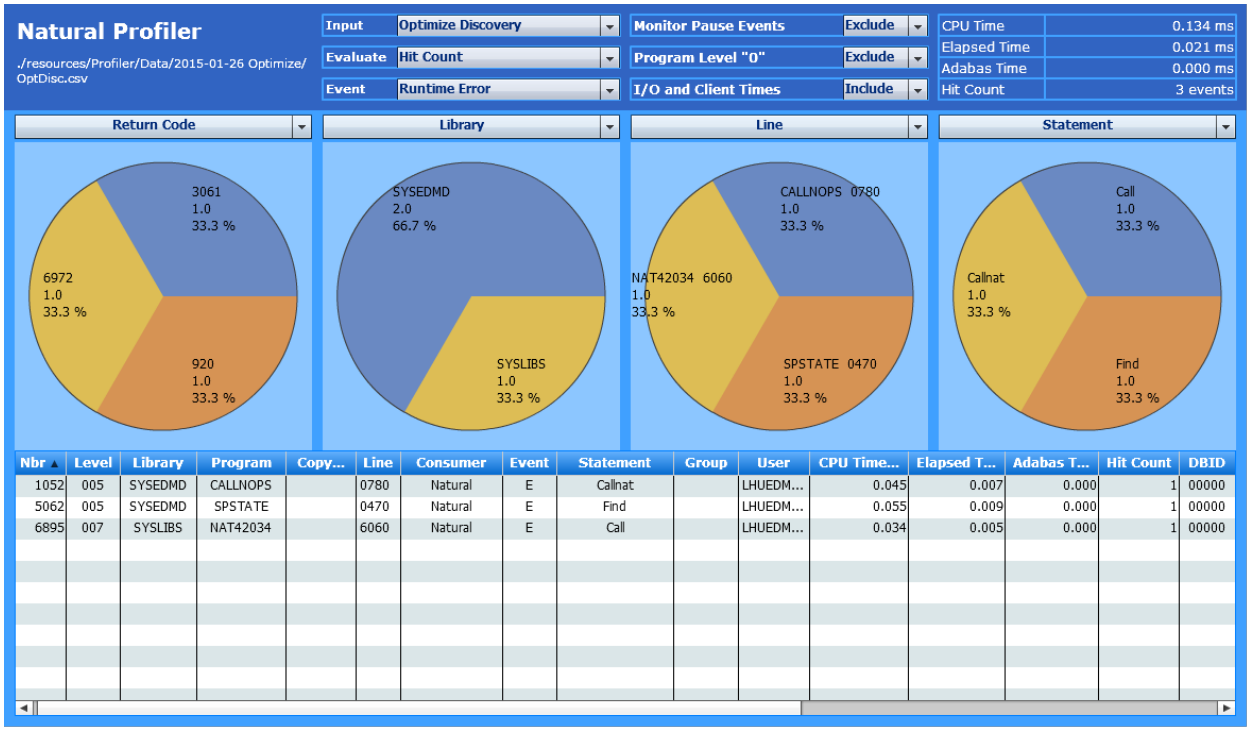
#### How many runtime errors occurred?

Use the following selections to find out:

**Evaluate:** Hit Count

**Event:** Runtime Error

**Criteria:** Return Code, Library, Line, Statement




In the example above, the **Hit Count** in the page header indicates that three runtime errors occurred during application execution. The charts show which errors occurred, the library, program and line where they occurred, and the statements that caused the errors.

# XXI SYSRDC Utility

# 107 SYSRDC Utility


- Functional Components of SYSRDC ..... 934
- Data-Collecting Events ..... 935
- Data Collected ..... 937
- Activating and Controlling the Natural Data Collector ..... 939
- Trace Recording ..... 940
- User Exits for External Monitoring/Accounting ..... 941
- Calling the CMRDC Interface ..... 942
- Example Programs ..... 946

The SYSRDC utility is used to record monitoring and accounting data on the internal process flow within a Natural application. This data can be used for evaluating Natural session activities in external or Natural programs. The data is collected at events performed within Natural.

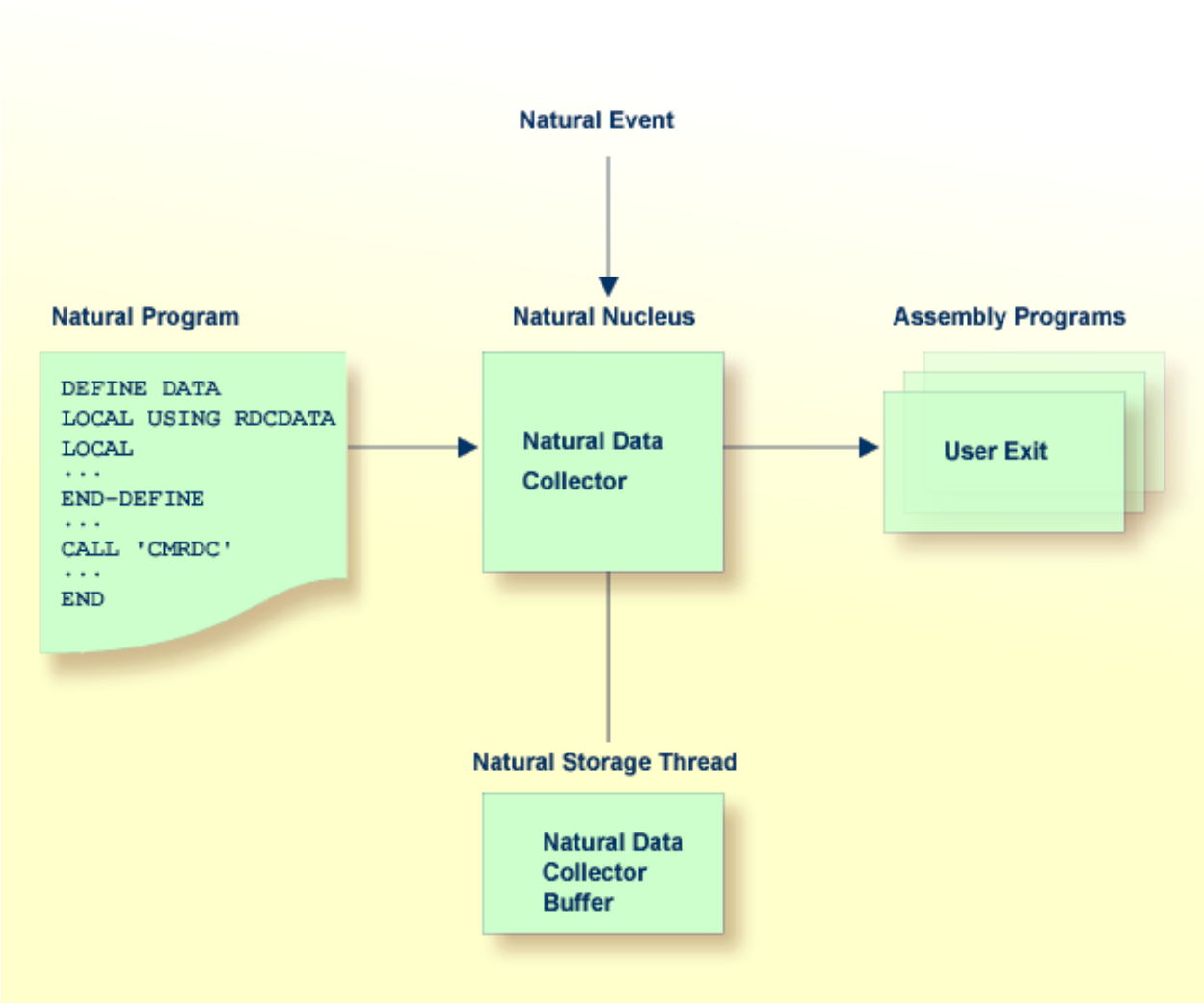
**Related Topics**

- See also the documentation on the *Profiler Utility*, a menu-driven tool for generating and listing trace records.
- *RDC - Configure the Natural Data Collector*

## Functional Components of SYSRDC

This section describes the components the SYSRDC utility provides in order to collect data and supply it for further processing.

- **Natural Data Collector as part of the Natural nucleus:**
  Collects data and controls data recording in the Natural Data Collector buffer.
- **User exits:**
  Passes data to external monitoring and accounting programs.
- **CMRDC interface:**
  Evaluates data in a Natural program for the current Natural session.




## Data-Collecting Events

Events specify activities within Natural. The table below lists the events at which data is collected within Natural and the types of event available.

Each type of event is assigned a one- or two-letter code where the first letter represents the group of event and any second letter the subtype.

For example, in the event type **PL, P** represents the event group “program” and **L** the subtype “load”.

| Event | Event Type |
|---|---|
| Session initialization. | SI |
| Session termination. | ST |
| Program load. | PL |
| Program start. | PS |
| Program termination. | PT |
| Before database call. | DB |
| After database call. | DA |
| Before terminal I/O. | IB |
| After a terminal I/O. | IA |
| Before non-Natural program call. | CB |
| After non-Natural program call. | CA |
| Runtime error. | E |
| Internal trace call.<br>**Note:** The information internal traces provide is only intended for debugging purposes by Software AG support. See **below**. | N |
| Natural statement.<br>**Note:** See **below**. | NS |
| Outbound RPC message. | RO |
| Inbound RPC message. | RI |
| Start of RPC request execution. | RS |
| Wait for RPC client request | RW |
| User-defined event. | U |

### To activate an internal trace call event

1 Set the Natural profile parameter `ITRACE=ON`.

2 Define the Natural components that are to issue internal trace calls:

- By using the NTTRACE macro of the Natural parameter module.

  Or, by setting the dynamic Natural profile parameter TRACE.

### To activate a Natural statement event

- Set Natural profile parameter `ITRACE=ON` and `TRACE=NATPROX`.

The NTTRACE macro and the parameters ITRACE and TRACE are described in the *Parameter Reference* documentation.

## Data Collected

The data the Natural Data Collector collects at events are described in the following section.

For the layout of the data, see the Natural source data set (file) NAMRDC, or the local data area RDCDATA that is provided in the Natural system library SYSRDC.

The data collected can be split into two categories: general and event-specific data:

- General Data
- Event-Specific Data

### General Data

The following general data is collected at every event:

- Software AG product name
- Product version
- Operating system
- TP monitor
- Run modes, such as addressing or residence mode of the Natural nucleus and the buffer pool
- TP user or batch job name
- TP terminal ID
- Current Natural user ID
- Current Natural Security user group ID
- Current Natural library
- Current Natural program
- Current program level
- Line number of program statement currently executed
- session CPU time (same format as `*CPU-TIME`)

## Event-Specific Data

The following data is only collected at the following events:

| Event | Data Elements |
|---|---|
| Session initialization | none |
| Session termination | Termination return code<br>Natural termination message code NAT99*nn*<br>Name of back-end program |
| Program load | Name of program to be loaded<br>Name of load library<br>Invocation type |
| Program start/termination | Program type<br>Program name<br>Program library name<br>Database ID of program library<br>File number of program library |
| Database call | Database type<br>Command code<br>Command ID<br>Database ID<br>File number<br>Pointer that lists parameter addresses (only useful for user exits)<br>Response code (only for event type DA)<br>Error subcode (only for event type DA)<br>Adabas command time (only for event type DA) |
| Terminal I/O | Number of bytes sent<br>Number of bytes read<br>Total session storage allocated<br>Compressed session storage length |
| Non-Natural program call | Name of program called<br>Calling mode, such as dynamic or static mode<br>Program link location<br>Parameter type<br>Parameter address<br>Program entry address<br>Response code (only for event type CA) |
| Runtime error | Natural system error message code<br>External abend code<br>Name of error handling program as specified with the Natural profile parameter ETA (see the relevant section in the *Parameter Reference* documentation). |
| Internal trace call | Up to 250 bytes of information on Natural nucleus components<br>The information provided is only intended for debugging purposes by Software AG personnel. |
| Natural statement | Up to 250 bytes of information on performed Natural statements |

| Event | Data Elements |
|---|---|
| RPC request | RPC-specific information:<br><br>Environment: C = client, S = server.<br><br>RDC subtype S:<br>subprogram name,<br>Adabas user ID (ETID),<br>conversation status,<br>logon indicator (Y = logon performed),<br>impersonation indicator of RPC request (Y = impersonation performed).<br><br>RDC subtypes O, I and W:<br>transport protocol,<br>RPC function,<br>type of client user ID,<br>length of message,<br>return code of transport layer,<br>external conversation ID,<br>client user ID,<br>server node (outbound message of client only),<br>server name (outbound message of client only). |
| User-defined event | Up to 250 bytes of user-defined information |

## Activating and Controlling the Natural Data Collector

The Natural Data Collector is activated and controlled by the Natural profile parameter RDC (see also the relevant section in the *Parameter Reference* documentation).

By default, the parameter is set to `RDC=OFF`, which means that the Natural Data Collector is deactivated and trace recording must be activated separately. To alter settings, use the *NTRDC macro* described in the RDC entry in *Parameter Reference*.

The following values and keyword subparameters are available for the profile parameter RDC:

| Value / subparameter | Description |
|---|---|
| ON | With ON, the trace recording is started automatically during Natural session start. |
| OFF | Natural Data Collector is deactivated and trace recording must be activated separately. |
| EVENT | Specify events to be recorded. Specify `ALL` to record all events. See *EVENT - Natural Data Collector Events to be Recorded*.<br><br>**Note:** As an alternative, you can use Natural statement `CALL 'CMRDC' 'T'`, see ***Selecting Event Types for Trace Recording***. |
| EXIT | Define user exits, see *EXIT - Define Natural Data Collector User Exits*. |

| Value / subparameter | Description |
|---|---|
| FNAT | Control trace recording while Natural system file programs are executing. See *FNAT - Trace Recording in the Natural System File*. |
| SIZE | Specify the buffer size for the Natural Data Collector. In addition, it controls the trace recording function of the Natural Data Collector. See *SIZE - Size of the Natural Data Collector Buffer*.<br><br>**Note:** As an alternative,you can use the equivalent Natural profile parameter RDCSIZE. |

## Trace Recording

The event data collected is always supplied to **user exits** to be used by external monitoring/accounting programs as described in the relevant section. Independently of the user exists, you can record the event data of the current Natural session in the Natural Data Collector buffer. This can be useful for testing purposes. In this section, the recording of data in the Natural Data Collector buffer is referred to as trace recording.

By default, trace recording is not active when you start a session. To make trace recording being activated when starting a session, set `RDC=ON`, see *RDC - Configure the Natural Data Collector*. As a prerequisite, the RDC subparameter SIZE must be set to a value greater than 2. You can also enter one of the following commands to start trace recording:

- system command `RDC ON`
- terminal command `%<RDC+`
- Natural statement `CALL 'CMRDC' 'S'`

**Notes:**

1. The Natural Data Collector buffer is filled in wrap-around mode; that is, the oldest record is overwritten when the buffer becomes full. At the end of the session, the contents of the buffer are deleted.
2. By default, trace recording does not cover system file FNAT. To include system file FNAT, set subparameter `FNAT=ON`.

Trace recording ends:

- system command `RDC OFF`
- terminal command `%<RDC-`
- Natural statement `CALL 'CMRDC' 'P'`
- when a session is terminated

## User Exits for External Monitoring/Accounting

The event data can be passed to external monitoring and accounting programs for evaluation of activities in Natural sessions. This is accomplished with the user exits and the examples of user-exit programs provided. User-exit programs are written in assembly language.

The Natural Data Collector supports any number of user exits. A user exit can be defined by the Natural profile parameter RDCEXIT (see the relevant section in the *Parameter Reference* documentation). An external monitoring/accounting program can be attached to each user exit.

Three exit names are predefined: RDCEX1, RDCEX2 and RDCEX3. If you use one of these names as entry point for your exit (which is linked to the Natural nucleus), the Natural profile parameter RDCEXIT is not required.

At every event listed under ***Data-Collecting Events***, the user exits take over control by using the standard linkage call conventions described in the table below:

| Register | Contents |
|---|---|
| 1 | Points to a parameter address list that consists of two addresses: one address points to the general data and the other points to event-specific data. The layout of these areas is mapped by the DSECT RDCGDATA and RDCLDATA respectively. Both DSECTs are supplied in source form in the Natural macro NAMRDC. |
| 13 | Points to a 72-byte standard save area. |
| 14 | Contains the return address. |
| 15 | Contains either the entry point address or the return code of the user exit. |

**Note:** The user exits are called independently of the CMRDC interface.

A user-exit program must have the same attributes as Natural; that is, it must have the same addressing mode, and it has to be reentrant. It must be linked with the Natural nucleus according to the conventions of statically linked non-Natural programs. See also the profile parameters CSTATIC and RCA described in the *Parameter Reference* documentation.

By default, a 400-byte exclusive work area (per session) is supplied for each user exit (the field RDCGWRKA). If a larger work area is required for a user exit, it can be specified after the user-exit name in the Natural profile parameter RDCEXIT (see the relevant section in the *Parameter Reference* documentation). The work-area length is passed on to the user exits in the field RDCGWRKL and can be used for verification. The location of this work area can change during a session due to Natural relocation, but its contents are preserved.

In TP-monitor environments, the TP anchor address is supplied (the field RDCGANCH); for example, the CSA address under CICS. It can be used to access system information.

If a Program Check occurs during the execution of a user-exit program, further data collecting is disabled for the rest of the session to avoid recursive abend situations.

The section below contains information on:

- Return Codes
- Examples of User-Exit Programs

### Return Codes

Non-zero return codes are only supported for two events:

- Before a database call, where Register 15 can contain an Adabas response code, which is stored in the control block; the Adabas call will not be executed.
- At program start, where Register 15 can contain a Natural error message code; the program will not be executed, but an error condition will occur with the specified number.

### Examples of User-Exit Programs

The following examples of user-exit programs are provided in the following Natural source data sets (files):

| Program | Data Set (File) | Used for |
|---|---|---|
| NAMRDC | NAT*nnn*.SRCE | DSECT macro for general and event-specific data |
| XNATRDC1 | NAT*nnn*.SRCE | Natural TSO Interface and Natural batch interface under operating systems z/OS and z/VSE |
| XNCFRDC1 | NCF*nnn*.SRCE | Com-plete |
| XNCIRDC1 | NCI*nnn*.SRCE | Natural CICS Interface |

## Calling the CMRDC Interface

The CMRDC application programming interface is used for retrieving and controlling the trace data recorded in the Natural Data Collector buffer. This can be useful for testing purposes.

**To invoke the CMRDC interface**

- In the Natural program, issue a CALL statement.

The functions provided by the CMRDC interface and the syntax that applies to the corresponding Natural CALL statements are described in the following section.

In addition, the Natural system library SYSRDC provides example programs and the appropriate local data area RDCDATA in which the structure of trace records is defined. Acording to this definition, a trace record has two main components: one to cover general data and one to cover event specific data.

The section below contains information on:

- Retrieving Trace Records
- Stopping and Restarting Trace Recording
- Selecting Event Types for Trace Recording
- User-Defined Events
- Retrieving the Trace Status
- CMRDC Return Codes

### Retrieving Trace Records

To read the data from the Natural Data Collector buffer, invoke CMRDC with the following Natural statement:

```
CALL 'CMRDC' function event-time gen-data event-data seq.-number
```

The following parameters are passed:

| Parameter | Format/Length | Explanation |
|---|---|---|
| *function* | A1 | F Get first trace record.<br>G Get next trace record.<br>N Get record of sequence number specified. |
| *event-time* | N10<br>or<br>N12 | Time of event<br>The time is passed back in either of the following formats:<br>*HHMMSSXXXX* *HH* = hours, *MM* = minutes, *SS* = seconds, *XXXX* = fraction of a second<br>*HHMMSSXXXXXX* *HH* = hours, *MM* = minutes, *SS* = seconds, *XXXXXX* = fraction of a second |
| *gen-data* | A252 | General data (field GDATA) |
| *event-data* | A252 | Event-specific data (field LDATA) |
| *seq.-number* | I4 | Sequence number of record<br>Only applies to Function **N** (see above). |

The retrieval functions stop recording data in the Natural Data Collector buffer implicitly.

### Stopping and Restarting Trace Recording

To stop or restart trace recording data in the Natural Data Collector buffer, invoke CMRDC with the following statement:

```
CALL 'CMRDC' function
```

The following parameter is passed:

| Parameter | Format/Length | Explanation |
|---|---|---|
| *function* | A1 | S Clear Natural Data Collector buffer and restart trace recording.<br>P Stop trace recording. |

### Selecting Event Types for Trace Recording

By default, all events are selected for trace recording. Use this function if you wish only specific events to be recorded.

**Notes:**

1. This function only selects the events at which data is to be recorded in the Natural Data Collector buffer. It does not affect the data passed to the user exits. Neither does it affect the status (started/stopped) of trace recording.
2. Events can also be specified by subparameter EVENT of the profile parameter RDC, see ***Activating and Controlling the Natural Data Collector***.
3. This function replaces any previous event definition.

To select the types of event to be recorded, invoke CMRDC with the following statement:

```
CALL 'CMRDC' function type ...
```

The following parameters are passed:

| Parameter | Format/Length | Explanation |
|---|---|---|
| *function* | A1 | T Select events for trace recording. |
| *type* | A1, A2 or A3 | The one- or two-letter code for the **event type** to be recorded as listed in *Data-Collecting Events*.<br><br>Specify any number of parameters for the event types desired. |

| Parameter | Format/Length | Explanation |
|---|---|---|
| | | Alternatively, you can select a range of events or no event:<br><br>ALL All event types.<br>*value** All event types that start with *value*.<br>For example, **P*** selects all events of the type program: PL, PS and PT.<br>blank A blank character selects no event. |

### User-Defined Events

To specify a user-defined event, invoke CMRDC with the following statement:

```
CALL 'CMRDC' function record
```

The following parameters are passed:

| Parameter | Format/Length | Explanation |
|---|---|---|
| *function* | A1 | U User-defined trace event. |
| *record* | A*nnn* | Trace record with a length (*nnn*) of up to 250 bytes |

### Retrieving the Trace Status

To retrieve the trace status and the event types currently set, invoke CMRDC with the following statement:

```
CALL 'CMRDC' function status types
```

The following parameters are passed:

| Parameter | Format/Length | Explanation |
|---|---|---|
| *function* | A1 | C Retrieve trace status. |
| *status* | A1 | Current trace status:<br><br>S Trace active<br>P Trace inactive |
| *types* | A*nn* | The two-letter codes for the **event types** currently set; see ***Selecting Event Types for Trace Recording***. |

| Parameter | Format/Length | Explanation |
|---|---|---|
| | | The variable *types* requires a minimum length (*nn*) of 64 bytes to provide sufficient space for all event types.<br><br>This parameter is optional. |

### CMRDC Return Codes

| Code | Meaning |
|---|---|
| 0 | Function executed successfully. |
| 4 | Last trace record. If event type is 0, there is no trace data for function **F**.<br>Only applies to Functions **F** and **G** (see above). |
| 8 | Too few parameters for this function. |
| 12 | Invalid function code. |
| 16 | Natural Data Collector not active, for example, RDCSIZE=0. |
| 20 | Natural Data Collector disabled after an error. |
| 24 | No buffer space available for trace recording (RDCSIZE=2). |
| 28 | Invalid parameter value.<br>Only applies to Functions **T** and **N** (see above). |

## Example Programs

The Natural system library SYSRDC contains the following example programs:

| Program | Function |
|---|---|
| RDCDISP | Display all records in the Natural Data Collector buffer and show the fields as specified in the program. See also ***Example Output of Program RDCDISP*** below. |
| RDCSTART | Restart trace recording. |
| RDCSTOP | Stop trace recording. |
| RDCSET | Select events for trace recording. |
| RDCUSER | User-defined event. |
| RDCSTAT | Retrieve trace status. |
| RDCDISIO | Display all I/O-related records in the Natural Data Collector buffer and show the fields as specified in the program. See also ***Internal Validity Check and Tracing for I/O-related Statements*** and ***Example Output of Program RDCDISIO*** below. |

### Example Output of Program RDCDISP

The example screen below shows the extract of an output report produced by the example program RDCDISP.

```
 ETIME1   TY  GCUID   LV   GPGM   T  GCAPL    PRLIB    PRNAM   CO  DB   FN  RC
--------- -- -------- -- -------- - -------- -------- -------- -- ---- ---- ---

   0.0000 SI SAG                    SYSTEM
   0.0002 DB SAG                    SYSTEM                     OP   10
   0.0002 DA SAG                    SYSTEM                     OP   10
   0.0000 DB SAG                    SYSTEM                     S1   10 1640
   0.0005 DA SAG                    SYSTEM                     S1   10 1640
   0.0001 PL SAG                    SYSTEM   SYSLIB   MAINMENU
   0.0001 PS SAG       1 MAINMENU F SYSTEM   SYSLIB   MAINMENU      10 1640
   0.0001 PL SAG       1 MAINMENU   SYSTEM   SYSLIB   NAT00029
   0.0000 DB SAG       1 MAINMENU   SYSTEM                     S1   10 1640
   0.0002 DA SAG       1 MAINMENU   SYSTEM                     S1   10 1640
   0.0000 PS SAG       2 NAT00029 N SYSTEM   SYSLIBS  NAT00029      10 1640
   0.0000 CB SAG       2 NAT00029   SYSTEM            CMMPP    S
   0.0000 CA SAG       2 NAT00029   SYSTEM            CMMPP    S
   0.0000 PT SAG       2 NAT00029 N SYSTEM   SYSLIBS  NAT00029
   0.0000 PL SAG       1 MAINMENU   SYSTEM   SYSLIB   USR2003P
   0.0000 PS SAG       2 USR2003P N SYSTEM   SYSLIB   USR2003P      10 1640
   0.0001 PT SAG       2 USR2003P N SYSTEM   SYSLIB   USR2003P
   0.0000 CB SAG       1 MAINMENU   SYSTEM            CMUB     S
```

The following table describes the columns that are displayed on the example screen, and the variables to which they refer. For some events the data displayed PRLIB etc. has a different meaning. For further information, see the comments in the program source and the local data area RDCDATA. For an explanation of the Natural system variables mentioned, refer to the *System Variables* documentation.

| Column | Explanation |
|---|---|
| ETIME1 | Time interval in seconds between the execution of the current and the previous event. |
| TY | Type of event as listed in ***Data-Collecting Events***. |
| GCUID | Current Natural user ID as assigned by the Natural system variable *USER. |
| LV | Program level. |
| GPGM | Name of the current program as assigned by the Natural system variable *PROGRAM. |
| T | Type of program. |
| GCAPL | ID of the current application library as assigned by the Natural system variable *APPLIC-ID. |
| PRLIB | ID of the library where the program is stored.<br>Applies to events of the type program, for example, event type PL. |
| PRNAM | Name of the program to be loaded for the type of event. |
| CO | Database command. |

| Column | Explanation |
|---|---|
| DB | Database ID. |
| FN | File number of Database. |
| RC | Database or program call response code. |

### Internal Validity Check and Tracing for I/O-related Statements

To detect `NAT1132` errors, and to facilitate debugging, information relevant to the error type (such as lengths of lines and fields, or buffer offsets) can be collected by **trace recording**. After processing an I/O-statement (such as `write`, `display` etc.) a validation routine is executed to check the buffers `IOPAGE` and `IOPATTR` for validity and consistency, for example with respect to presence of a line separator. If a potential error is detected, a message (`NAT1132 condition detected`) is written to the Natural Data Collector buffer, together with information that helps to locate the error, and trace recording is stopped to avoid this result to be hidden by subsequent irrelevant data.

To activate NAT1132 trace recording, specify the following Natural profile parameters:

```
ITRACE=ON,RDCSIZE=128,TRACE=IO
```

If a trace was created for a program that you have run, you can display I/O-related records by executing `RDCDISIO` in system library SYSRDC.

#### Example Output of Program RDCDISIO

The following screen shows (a fragment of) a trace report produced by the program `RDCDISIO`. See the definition list below for an explanation of the column headings.

```
 GCUID   LV   GPGM    GCAPL                       NDATA
-------- -- -------- -------- ---------------------------------------------
MK        1 LOGON    SYSTEM   INO2  08 F 005041700618 (R6)
MK        1 MYDEMO   MK       INO2  00 F 001A00200609 (R6)
MK        1 MYDEMO   MK       IOIPR    F 001A00200609 50 50 0050 0050
MK        1 MYDEMO   MK       INO2  04 F 001A00200609 (R6)
MK        1 MYDEMO   MK       INO2  00 F 001A00200609 (R6)
MK        1 MYDEMO   MK       IOIPR    F 001A00200609 50 50 0050 0050
MK        1 MYDEMO   MK       INO2     F 10C000C8C9C5 (R6) BEG OF  IOPRI
MK        1 MYDEMO   MK       IOEUP    F 000000A0 0F  (PATTRADF/LNG)
MK        1 MYDEMO   MK       IOEUP    F 000000A0 0F  (PATTRADF/LNG)
MK        1 MYDEMO   MK       IODRU    F 000000B0 3F  (PATTRADF/LNG)
MK        1 MYDEMO   MK       INO2     F 000A00301804 (R6) END OF  IOPRI
MK        2 MYDEMON  MK       INO2  00 N 001801700609 (R6)
MK        2 MYDEMON  MK       IOIPR    N 001801700609 4F 50 004F 0050
MK        2 MYDEMON  MK       IOIPR    N LINE SIZE HAS CHANGED
MK        2 MYDEMON  MK       INO2     N 058000800104 (R6) BEG OF  IOPRI
MK        2 MYDEMON  MK       IOEUP    N 000000F0 08  (PATTRADF/LNG)
MK        2 MYDEMON  MK       IOEUP    N 000000F0 08  (PATTRADF/LNG)
MK        2 MYDEMON  MK       IOEUP    N 000000F9 32  (PATTRADF/LNG)
MK        2 MYDEMON  MK       IOEUP    N 000000F9 32  (PATTRADF/LNG)
MK        2 MYDEMON  MK       IOEUP    N 0000012C 12  (PATTRADF/LNG)
MK        2 MYDEMON  MK       IOEUP    N 0000012C 12  (PATTRADF/LNG)
MK        2 MYDEMON  MK       IOPRI    N NAT1132 CONDITION DETECTED
MK        2 MYDEMON  MK       IOEUP    N 0000013F 32  (PATTRADF/LNG)
MK        2 MYDEMON  MK       IOEUP    N 0000013F 32  (PATTRADF/LNG)
MK        2 MYDEMON  MK       IOEUP    N 00000172 32  (PATTRADF/LNG)
MK        2 MYDEMON  MK       IOEUP    N 0000018F 1E  (PATTRADF/LNG)
MK        2 MYDEMON  MK       IOEUP    N 0000018F 1E  (PATTRADF/LNG)
MK        2 MYDEMON  MK       IODRU    N 000001AE 30  (PATTRADF/LNG)
MK        2 MYDEMON  MK       INO2     N 00060180FF08 (R6) END OF  IOPRI
MK        2 MYDEMON  MK       INO2     N NAT1132 CONDITION 1 DETECTED
```

| | |
|---|---|
| GCUID | Current Natural user ID as assigned by the Natural system variable `*USER`. |
| LV | Program level. |
| GPGM | Name of the current program as assigned by the Natural system variable `*PROGRAM`. |
| GCAPL | ID of the current application library as assigned by the Natural system variable `*APPLIC-ID`. |
| NDATA | Up to 250 bytes of information on Natural nucleus components, to be used by Software AG personnel for debugging. |

# XXII SYSRPC Utility

The utility SYSRPC is used to maintain remote procedure calls on the client side.


**Invoking and Terminating SYSRPC**

**Service Directory Maintenance**

**Replacing Items in the Service Directory**

**Generating Interface Objects - General Considerations**

**Generating Single Interface Objects with Parameter Specification**

**Generating Multiple Interface Objects**

**Calculating Size Requirements**

**Parameter Maintenance**

**Server Command Execution**

**Listing Servers Registered on EntireXBroker**

**Overview of SYSRPC Direct and Batch Commands**


**Related Topics:**

- For information on how to apply the SYSRPC utility functions to establish a framework for communication between server and client systems, refer to the *Natural Remote Procedure Call (RPC)* documentation.
- For explanations of expressions relevant to the SYSRPC utility, see also the section *Natural RPC Terminology* in the *Natural RPC (Remote Procedure Call)* documentation.
- The use of SYSRPC can be controlled by Natural Security: see *Protecting Utilities* in the *Natural Security* documentation.
- For information on Application Programming Interfaces provided to maintain remote procedure calls, see *Application Programming Interfaces for Use with Natural RPC* in the *Natural RPC (Remote Procedure Call)* documentation.
- For detailed information regarding EntireX Broker features and components, refer to the appropriate *EntireX Broker* documentation.

# 108 Invoking and Terminating SYSRPC


- Invoking SYSRPC ..... 954
- Terminating SYSRPC ..... 955
- Invoking Online Help ..... 955

This section provides instructions for starting and terminating the SYSRPC utility and invoking the help function.

## Invoking SYSRPC

You can invoke the SYSRPC utility by using either a system command or menu functions.

### To invoke SYSRPC using a system command

- In the Command line, enter the following command:

  ```
  SYSRPC
  ```

  and choose ENTER.

  The **Client Maintenance** menu of the SYSRPC utility appears.

### To invoke SYSRPC using menu functions

1. From the Natural **Main Menu**, select **Maintenance and Transfer Utilities** and choose ENTER.

   The **Maintenance and Transfer Utilities** menu is displayed.
2. Select **Maintain Remote Procedure Calls** and choose ENTER.

   The **Client Maintenance** menu of the SYSRPC utility appears.

From the **Client Maintenance** menu, you can invoke all functions available for RPC (remote procedure call) maintenance:

- **Service Directory Maintenance**
- **Generating Interface Objects - General Considerations**
- **Parameter Maintenance**
- **Server Command Execution**

See the relevant sections for descriptions of these functions.

## Terminating SYSRPC

### To terminate the SYSRPC utility

- In the **Code** field of the **Client Maintenance** menu, enter a period (.).

  Or:

  Choose PF3 (Exit).

## Invoking Online Help

### To invoke the online help function

- Choose PF1 (Help).

# 109 Service Directory Maintenance


- Service Directory Concept ..... 958
- Invoking Service Directory Maintenance ..... 959
- Fields on the Service Directory Screen ..... 961
- Commands for Service Directory Maintenance ..... 964

The **Service Directory Maintenance** function is used to maintain a service directory in order to connect the client's calling program to a subprogram on a server.

The service directory information is stored in the NATCLTGS subprogram in the library that is defined with the `NTRPC/RPC` keyword subparameter `RPCSDIR` (see the *Parameter Reference* documentation). If `RPCSDIR` is set, the **Service Directory Maintenance** function references the library specified with `RPCSDIR`. If `RPCSDIR` is not set (this is the default), the library where you are logged on is referenced. In this case, log on to the library (or one of its steplibs) used by the client at runtime before you perform the **Service Directory Maintenance** function.

The name of the library referenced for service directory maintenance is indicated in the upper right corner of the **Service Directory** screen (see *Invoking Service Directory Maintenance*). If `RPCSDIR` is set, the screen title contains **Central**, which indicates that the library displayed on the screen is *not* the library where you are currently logged on, but the central library specified with `RPCSDIR`.

**Attention:**

If NATCLTGS is stored in the Natural system library SYSRPC, we strongly recommend that you move NATCLTGS to the application library (or one of its steplibs) used by the client.

For further information on how to apply the **Service Directory Maintenance** function, refer to *Specifying RPC Server Addresses* described in *Operating a Natural RPC Environment* in the *Natural RPC (Remote Procedure Call)* documentation.

**API for Service Directory Maintenance Functions:**

You can use the application programming interface (API) `USR8216N` to perform service directory maintenance functions. `USR8216N` retrieves an existing service directory and adds, changes or deletes entries in the service directory. `USR8216N` is supplied in the Natural SYSEXT system library. For handling instructions, see *Using a Natural API* in the section *SYSEXT Utility*.

## Service Directory Concept

A service directory has a hierarchical structure with a cascading list to assign subordinate to superior fields. The highest hierarchical level is node and the lowest is program. You cannot enter node, server, library and program in the same line. If you do so, an appropriate error message appears. You need to enter the value of a subordinate field in the lines below the superior field. You can assign several servers to a node, several libraries to a server and several programs to a library.

- Nodes and Servers

### Nodes and Servers

In *Example 1 - Standard View of Service Directory*, two servers are defined for one node. Both servers are connected to the same node: `ETB045`. The remote `CALLNAT` to subprogram `SUB1` is executed on server `NRPC001`, whereas subprograms `SUB2` and `SUB3` are executed on server `NRPC002`.

The server names specified here must be identical to the server names specified for the server with the `NTRPC/RPC` keyword subparameter `SRVNAME` described in the *Parameter Reference* documentation. Analogously, the node name in the service directory must be identical to the node name specified for the server with the `NTRPC/RPC` keyword subparameter `SRVNODE` described in the *Parameter Reference* documentation.

## Invoking Service Directory Maintenance

**Attention:**

The **Service Directory Maintenance** function invokes the Natural editor. As a result, data stored in the source work area may be lost when invoking **Service Directory Maintenance**. An appropriate message will warn you not to delete any existing entries unintentionally: choose PF12 to cancel the function or choose ENTER to confirm the action and clear the source work area.

#### To invoke the Service Directory Maintenance function

1 In the **Code** field of the **Client Maintenance** menu, enter the following command:

```
SM
```

2 Choose ENTER.

- If the service directory already contains service definitions, a window appears with the following message:

  ```
  Existing service definitions found
  ```

  In the **Code** field of the window, enter an `A` (default) to keep old definitions and append new ones and choose ENTER.

  Or:
  In the **Code** field of the window, enter an `I` to ignore all existing definitions and delete them from the service directory and choose ENTER.

The standard view of the **Service Directory** screen is displayed as shown in the following example:

**Example 1 - Standard View of Service Directory**

```
15:00:11            ***** NATURAL SYSRPC UTILITY *****              2016-07-18
                             - Service Directory -             Library SAGTST

             Node         Tr.        Server      Logon  Library     Program
1      ETB045__________    B    ________________   _   ________    ________
2      ________________    _    NRPC001_________   N    ________    ________
3      ________________    _    ________________   _    SYSTEM__    ________
4      ________________    _    ________________   _    ________    SUB1____
5      ________________    _    NRPC002_________   Y    ________    ________
6      ________________    _    ________________   _    SYSTEM__    ________
7      ________________    _    ________________   _    ________    SUB2____
8      ________________    _    ________________   _    ________    SUB3____
9      ________________    _    ________________   _    ________    ________
10     ________________    _    ________________   _    ________    ________
11     ________________    _    ________________   _    ________    ________
12     ________________    _    ________________   _    ________    ________
13     ________________    _    ________________   _    ________    ________
14     ________________    _    ________________   _    ________    ________
15     ________________    _    ________________   _    ________    ________
16     ________________    _    ________________   _    ________    ________

Command ===>


Enter-PF1---PF2---PF3---PF4---PF5---PF6---PF7---PF8---PF9---PF10--PF11--PF12---
      Help  Long  Exit  Find  -H    +H    -P    +P    Top   Bot   <     Canc
```

The **Service Directory** screen provides a maximum of 500 lines for input.

3 If you choose PF11 or enter the less than (<) sign in the Command line, the extended node/server view of the **Service Directory** screen is displayed similar to the following example:

**Example 2 - Extended Node/Server View of Service Directory**

```
15:09:01            ***** NATURAL SYSRPC UTILITY *****              2016-07-18
                             - Service Directory -             Library SAGTST

                    Node                 Tr.              Server            Logon
1    ETB045_____________________________  B  ________________________________  _
2    __________________________________  _  NRPC001___________________________  N
3    __________________________________  _  ________________________________  _
4    __________________________________  _  ________________________________  _
5    __________________________________  _  NRPC002___________________________  Y
6    __________________________________  _  ________________________________  _
7    __________________________________  _  ________________________________  _
8    __________________________________  _  ________________________________  _
9    __________________________________  _  ________________________________  _
10   __________________________________  _  ________________________________  _
11   __________________________________  _  ________________________________  _
12   __________________________________  _  ________________________________  _
13   __________________________________  _  ________________________________  _
14   __________________________________  _  ________________________________  _
15   __________________________________  _  ________________________________  _
16   __________________________________  _  ________________________________  _

Command ===>


Enter-PF1---PF2---PF3---PF4---PF5---PF6---PF7---PF8---PF9---PF10--PF11--PF12---
      Help  Long  Exit  Find  -H    +H    -P    +P    Top   Bot   >     Canc
```

If you choose PF11 or enter the greater than (>) sign in the Command line, the standard view of the **Service Directory** screen is displayed as shown in *Example 1 - Standard View of Service Directory*.

## Fields on the Service Directory Screen

The **Service Directory** screen contains the following input fields (one entry per line):

| Field | Description |
|---|---|
| **Node** | The name of the node to which the remote `CALLNAT` is sent. See also *Natural RPC Terminology* in the *Natural RPC (Remote Procedure Call)* documentation.<br><br>The maximum length of input is as follows:<br><br>Standard view of the **Service Directory** screen: 16 characters<br>Extended node/server view of the **Service Directory** screen: 32 characters |

| Field | Description |
|---|---|
| | **Long Name** window (see PF2 in *Direct Commands and PF Keys*) of the **Service Directory** screen: up to 192 characters. |
| **Tr.** | The transport protocol:<br><br>B indicates EntireX Broker ACI protocol. |
| **Server** | The name of the server to which the remote CALLNAT is sent. See also *Natural RPC Terminology* in the *Natural RPC (Remote Procedure Call)* documentation.<br><br>The maximum length of input is as follows:<br><br>Standard view of the **Service Directory** screen: 16 characters<br>Extended node/server view of the **Service Directory** screen: 32 characters |
| **Logon** | Initiates a Natural logon to the server.<br><br>This is possible at server or node level and applies to all definitions made at a hierarchically lower level. If the **Logon** option has been set for a specific server, it applies to all associated library and subprogram definitions.<br><br>Possible values are as follows:<br><br>Y — If set to Y (Yes), for each non-conversational CALLNAT request or for each start of a conversation, the client initiates a Natural logon to the server using the current library name on the client, regardless of the libraries in the subordinate **Library** column that belongs to the **Server** field. You can use the Application Programming Interface USR4008N to specify a different library (see also *Logging on to a Different Library* in *Using the Logon Option*).<br><br>N or *blank* — If set to N (No) or if no value is entered, no logon is initiated.<br><br>After the remote CALLNAT has been executed (successfully or not) or at the end of a conversation, the server library is reset to its previous state. For more information, see *Using the Logon Option* in the *Natural RPC (Remote Procedure Call)* documentation.<br><br>See also ***Server Command Execution***. |
| **Library** | SYSTEM or the name of the library to which your client application is logged on during the execution of the remote CALLNAT. |
| **Program** | The name of the remote subprogram to be accessed from the client. |

| Field | Description | |
|---|---|---|
| | | |
| | You can enter a name or a range of names. Valid names are any combinations of one or more alphanumeric characters with one or more asterisks (*) and/or one or more question marks (?) where: | |
| | asterisk (*) denotes any string of characters,<br>question mark (?) denotes a single character. | |
| | | |
| | Invalid combinations are: | |
| | | |
| | `*?` | An asterisk followed by a question mark is converted to `?*`. |
| | `**` | Two or more consecutive asterisks are converted to a single asterisk. |

### Selection Criteria for Node and Server

At Natural runtime, the selection of a node and server depends on the value of the fields **Program** and **Library**. Comply with the following conditions:

**Non-conversational** `CALLNAT`

1. The **Library** field must contain the name of the current application library or SYSTEM.
2. The name of the subprogram specified in the `CALLNAT` statement must be contained in the **Program** field, which belongs to the **Library** field in point (1).

**Conversational** `CALLNAT`

1. The **Library** field must contain the name of the current application library or SYSTEM.
2. All subprograms specified in the `OPEN CONVERSATION` statement must be contained in a **Program** field, which belongs to **Library** field in point (1).

The node and server used for a non-conversational or conversational `CALLNAT` are taken from the superior **Node** and **Server** fields of the **Library** field in point (1).

# Commands for Service Directory Maintenance

This section contains information on the commands provided on the **Service Directory** screen:

- Line Commands
- Direct Commands and PF Keys

## Line Commands

The line commands provided on the **Service Directory** screen can be used to copy, move or delete single or multiple lines that contain field values.

Enter a line command at the beginning of a line, that is, overwrite the sequential number and choose ENTER.

See also *To copy or move a block of lines* and the direct command RESET.

| Line Command | Function |
|---|---|
| A | Copies or moves the block of lines marked with CC or MM below the line in which the command was entered. |
| CC | Marks the block of lines to be copied. |
| D | Deletes the marked line. |
| DD | Marks and deletes a block of lines.<br><br>Mark a block of lines by entering this command in the first and the last line of the block and choose ENTER to execute the command. |
| I | Inserts five empty lines below the line in which the command was entered. |
| MM | Marks the block of lines to be moved. |
| P | Copies or moves the block of lines marked with CC or MM above the line in which the command was entered. |

### To copy or move a block of lines

1 At the beginning of the line where the block starts, overwrite the sequence number with either of the following line commands:

```
CC
```

to copy the block or

```
MM
```

to move the block.

2 At the beginning of the line where the block ends, overwrite the sequence number with either of the following line commands:

```
CC
```

to copy the block or

```
MM
```

to move the block.

3 Choose ENTER.

The line commands disappear, the sequence numbers are displayed again and the block of lines has been marked.

4 At the beginning of the line below or above which you want to place the marked block of lines, enter either of the following line commands:

```
A
```

to copy or move the block *below* the specified line or

```
P
```

to copy or move the block *above* the specified line.

Note that you can only execute `A` or `P` on lines where at least one field is filled.

5 Choose ENTER.

The block of lines is copied or moved below or above the specified line.

### Direct Commands and PF Keys

The following direct commands can be entered in the Command line of the **Service Directory** screen and/or are provided as PF keys:

| Direct Command | PF Key | Function |
|---|---|---|
| EXPIRATION | | The remote directory data is loaded at runtime. The expiration time in seconds determines the period of validity of this data. If directory data is requested after the expiration time set, it will automatically be reloaded. If the expiration time is set to `0`, the remote directory data will not be reloaded.<br><br>With the direct command `EXPIRATION`, you can enter an expiration time in seconds, for example, `EXPIRATION 86400`. Maximum is an 8-digit number.<br><br>If you do not provide a parameter with the command, the **Expiration Time** window appears where you can display or modify the current time. |
| RESET | | Removes the line marks set with a line command as described in *Line Commands*. |

| Direct Command | PF Key | Function |
|---|---|---|
| | | Note that if lines have been marked incorrectly, an appropriate message occurs and you have to remove the erroneous line command before you enter `RESET`. |
| | PF1 | Invokes the editor online help. |
| | PF2 | Opens the **Long Name** window where you can enter a node name of up to 192 characters. |
| | PF3 | Exit. Prompts you to save modifications and exit the **Service Directory** screen. |
| `FIND` | PF4 | Invokes the **Find Item** window where you can search for a name:<br><br>**Find what** — Enter an alphanumeric search string of up to 32 characters.<br><br>**Case sensitive** — Replace the default setting `N` (No) by `Y` (Yes) to distinguish between uppercase and lowercase characters.<br><br>**Whole words only** — Replace the default setting `N` (No) by `Y` (Yes) to search for whole words only.<br><br>Choose ENTER to start searching and move from one hit to the next if one exists. Press PF4 to restart searching from the beginning.<br><br>The hits are marked with the cursor. |
| `REPLACE` | PF16 | Invokes the **Replace Item** window where you can search for and replace single or multiple names (not-case-sensitive):<br><br>**Find** — Enter an alphanumeric search string of up to 32 characters.<br><br>**Replace with** — Enter an alphanumeric replace string of up to 32 characters.<br><br>**Whole words only** — Replace the default setting `N` (No) by `Y` (Yes) to search for whole words only.<br><br>**Search only** — All names in the service directory are searched for matches by default (blank field entry).<br><br>You can enter one of the following letters to restrict the search to one of the following items:<br><br>`N` — Node names only<br>`S` — Server names only |

| Direct Command | PF Key | Function |
|---|---|---|
| | | L — Library names only<br>P — Program names only<br><br>**Replace all** — Replaces all occurrences of the search string found.<br><br>Choose ENTER to start searching and move from one hit to the next if one exists. Press PF4 to restart searching from the beginning.<br><br>The hits are marked with the cursor. |
| REPLACE *replace-clause* | | Performs the replace functions provided in the **Replace Item** window. It corresponds to the ***replace-clause*** of the SYSRPC SM REPLACE command. |
| -H | PF5 | Scrolls half a page backward/forward. |
| +H | PF6 | |
| -P | PF7 | Scrolls one page backward/forward. |
| +P | PF8 | |
| TOP | PF9 | Scrolls to the beginning of the list. |
| BOT | PF10 | Scrolls to the end of the list. |
| | PF11 | Toggles between the standard view of the **Service Directory** screen (see *Example 1 - Standard View of Service Directory*) and the extended view of the fields **Node** and **Server** (see *Example 2 - Extended Node/Server View of Service Directory*). |
| > | PF11 | Displays the extended view of the fields **Node** and **Server**. The extended node/server view does not display the fields **Library** and **Program** as shown in *Example 2 - Extended Node/Server View of Service Directory*. |
| < | PF11 | Displays the standard view of the **Service Directory** screen as shown in *Example 1 - Standard View of Service Directory*. |
| CANCEL | PF12 | Exits the **Service Directory** screen without saving any modifications. |

# 110 Replacing Items in the Service Directory


- Syntax of SYSRPC SM REPLACE ..... 970

You can use the `SYSRPC SM REPLACE` direct command to replace the names of nodes, servers, libraries and programs defined in a service directory.

`SYSRPC SM REPLACE` corresponds to the `SM REPLACE` command you can enter on the **Service Directory Maintenance** screen described in *Commands for Service Directory Maintenance*.

`SYSRPC SM REPLACE` can be used in online and batch mode.

This section contains information on:

## Syntax of SYSRPC SM REPLACE

```
SYSRPC SM REPLACE replace-clause
```

*replace-clause*

The *replace-clause* of the `REPLACE` command corresponds to the *replace-clause* of the `SM REPLACE` direct command.

```
[ ANY     ]
[ NODE    ]
[ SERVER  ] search-string WITH replace-string [ ALL   ] [WHOLE]
[ LIBRARY ]                                   [ FIRST ]
[ PROGRAM ]
```

The syntax items are explained in the following table:

| | |
|---|---|
| `ANY` | Searches for all names specified in the service directory.<br><br>This is the default value. |
| `NODE` | Searches for node names only. |
| `SERVER` | Searches for server names only. |
| `LIBRARY` | Searches for library names only. |
| `PROGRAM` | Searches for program names only. |
| *search-string* | An alphanumeric search string of up to 32 characters. |
| `WITH` | Introduces the *replace-string*. |
| *replace-string* | An alphanumeric replace string of up to 32 characters. |
| `ALL` | Replaces all occurrences of the search string found. |
| `FIRST` | Replaces only the first occurrence of the search string found.<br><br>This is the default value. |
| `WHOLE` | Replaces only occurrences that match the whole search string. |

**Note:** The search operation is not case-sensitive.

# 111 Generating Interface Objects - General Considerations

An interface object is a Natural subprogram that is used to connect the client's calling program to a subprogram on a server.

Interface objects are actually not required if automatic Natural RPC (Remote Procedure Call) execution is used with the one important exception described below. However, it can be advantageous to generate interface objects as explained in *Interface Objects and Automatic RPC Execution* in the section *Operating a Natural RPC Environment* in the *Natural RPC (Remote Procedure Call)* documentation.

**Note for EntireX RPC Servers:**
It is recommended to generate an interface object if you want to call an EntireX RPC server. In this case, you have to use the appropriate SYSRPC **Interface Object Generation** function described in this section to define the same group structure and attributes (parameter direction) as in the IDL (Interface Definition Language) definition of the subprogram. If the IDL does not contain group structures, it is recommended to use the direct command `COMPAT IDL` before generating the interface object. For details, refer to ***Special Considerations for Calling EntireX RPC Servers***.

**Note for Reliable RPC:**
It is recommended to generate an interface object if you want to use reliable RPC. If the parameter definitions do not contain group structures, you have to set `COMPAT IDL` before generating the interface object. (For details refer to ***Special Considerations for Reliable RPC***).

You can generate an interface object from new parameter definitions or from existing definitions in a subprogram or a parameter data area (PDA).

**Caution:** The subprogram or PDA used for generating an interface object can no longer be referenced in the local environment on the client side. The function **Interface Object Generation** completely changes the source of the subprogram so that it becomes unusable for local program calls.

The following sections describe the functions and commands provided to generate single or multiple interface objects:

- **Generating Single Interface Objects with Parameter Specification**
- **Generating Multiple Interface Objects**

# 112 Generating Single Interface Objects with Parameter Specification


- Using the Interface Object Generation Function ..... 976
- Specifying Parameters ..... 979
- Examples of Interface Object Generation ..... 981

The function **Interface Object Generation** provides the option to generate single interface objects online on a separate screen. You either type in the parameter definitions required or read them in from an existing subprogram or a parameter data area (PDA).

Generating an interface object from a PDA saves you the effort of creating a subprogram and defining an inline parameter data area.

## Using the Interface Object Generation Function

Interface objects are generated into the current Natural library in the current system file. Therefore, we strongly recommend that you log on to the application library (or one of its steplibs) used by the client at execution time of the remote `CALLNAT`.

**Important:** The function **Interface Object Generation** overwrites any data contained in the source work area. When you invoke the function, a corresponding message will warn you not to delete any existing data unintentionally: choose PF12 to cancel or choose ENTER to confirm the action and overwrite the contents of the source work area.

### To generate a single interface object

1 Before you invoke the SYSRPC utility, consider the following:

- Log on to the library into which you want to generate the interface object.
- If you generate an interface object from a PDA: Rename or copy the PDA from which you want to generate the interface object if there are objects on the client side that still reference this PDA. The new name of the PDA must be identical to the name of the remote `CALLNAT` program.

2 In the **Code** field of the **Client Maintenance** menu, enter the following command:

```
IG
```

3 Choose ENTER.

The **Generate Interface Object** screen appears.

4 In the **Program Name** field, enter the name of the interface object to be generated.

The name of the interface object must be identical to the name of the remote `CALLNAT` program.

The **Library** field is preset to the name of the current library and cannot be changed.

In the **Compression** field, enter compression type `0`, `1` or `2` (default is `1`); see *Using Compression* described in *Operating a Natural RPC Environment* in the *Natural RPC (Remote Procedure Call)* documentation.

5 Choose ENTER.

- If the name entered in the **Program Name** field corresponds to the name of an object that already exists in the assigned library, a window appears with an appropriate message:

  Enter an `N` (No) and choose ENTER if you want to cancel the operation. You will return to the **Client Maintenance** menu.

  Or:
  Enter a `Y` (Yes) and choose ENTER if you want to continue with generating interface objects.

  If the specified name is identical to a cataloged object of the type subprogram or PDA, the parameter definitions of the respective subprogram or PDA are displayed on the **Interface Generation** screen.

  If the specified name is identical to an already generated interface object for which also a source object exists, all field attributes (see also ***Specifying Parameters***) are retained. Otherwise, all field attributes are set to `M` (modifiable).

- If the string entered in the **Program Name** field, does *not* match the name of an object that already exists in the assigned library, an empty **Interface Object Generation** screen is displayed.

6 On the **Interface Object Generation** screen, add or modify the parameters to be used in the interface object as described in ***Specifying Parameters***.

The commands provided on the **Interface Object Generation** screen correspond to the commands described in ***Commands and PF Keys*** in the section *Service Directory Maintenance*.

Exceptions:

| Attribute | Values |
|---|---|
| EXPIRATION | Not applicable to interface object generation. |
| COMPAT | IDL Generate an interface object according to IDL requirements.<br>NONE Generate an interface object according to Natural requirements.<br>void Show COMPAT setting.<br>**Note:** See also: ***Special Considerations for Reliable RPC*** and ***Special Considerations for Calling EntireX RPC Servers*** . |
| LIMIT | 32000 Sets the upper size limit to 32000 bytes.<br>1GB Sets the upper size limit to 1 GB.<br>void Removes a size limit set with `LIMIT 32000` or `LIMIT 1GB`. |

7 Choose ENTER to generate the interface object and to exit. The interface object is generated into the assigned library.

The **SYSRPC - Information** window appears which indicates the size of the interface object requires for sending data from the client to the server or vice versa. The size includes internal RPC information used for the interface object. The indication of the size helps you configure the middleware layer used; for example, the Broker attribute file when EntireX Broker is used.

The following message appears in the **SYSRPC - Information** window when you generate an interface object from the example subprogram TESTS5 (see ***Example 1*** below):

```
Interface Object TESTS5 is generated in library SAGTEST (99,49).
   It requires:
        Send length: 2249 bytes
     Receive length: 2221 bytes
```

If dynamic parameters, X-arrays or X-group arrays are used, this message only indicates the minimum length requirements. The actual length requirements can only be determined during program execution and may be different from call to call. If the `Send length` or the `Receive length` exceeds the Entire Net-Work limit of 32000 bytes, a window appears with a corresponding warning:

Enter a `Y` (Yes) to continue, or an `N` (No) to cancel the generation. If you enter a `Y`, this setting is kept for the entire SYSRPC session, that is, you can continue generating interface objects without receiving further warnings.

If the total data (without internal RPC information) sent or received exceeds the limit of 1073739357 bytes (which is 1 GB minus 2467 bytes of internal RPC information), SYSRPC stops processing and issues a corresponding error message. This error message displays the subtotal of the data in bytes that could be transferred at the field up to which the subtotal was calculated. The corresponding field is then marked. In this case, reduce the amount of data before you continue the generation.

If the interface object was generated in the Natural system library SYSRPC, you have to move it to the application library or steplib using the Natural transfer utility SYSMAIN or the Object Handler. Note that you may have to recatalog the source of the interface object in the target environment.

## Specifying Parameters

In the input fields provided on the **Interface Object Generation** screen, you can enter the parameter definitions that are used in the interface object. You can specify a maximum of 5000 parameters. Unless indicated in the table below, input in the fields is mandatory.

| Field | Description |
|---|---|
| **Level** | The level of the field.<br><br>A level can be a number in the range from `01` (highest level) to `99` (lowest level). The leading `0` is optional.<br><br>See also ***Defining Groups*** and ***Example 2*** for an example of a group definition. |
| **Attr** | The attribute of the parameter:<br><br>`M` (modifiable - INOUT), `O` (output - OUT) or `I` (input - IN).<br><br>Parameters assigned a level number of `2` or greater are considered to be a part of a group. Parameters within a group must have the same attribute as the immediately preceding group that is assigned one level higher. For nested groups, this is the attribute of the group with the highest level. For an example of a group definition, see ***Example 2***.<br><br>If an interface object has been generated from a subprogram or from a PDA, the attribute is `M` by default, which may need modification.<br><br>If an interface object has been generated from another interface object, the attribute values specified for the original object are retained.<br><br>The generated interface object contains a comment that indicates the attribute specified for the parameter: `IN`, `OUT` or `INOUT`. |
| **Type** | A Natural data format such as `N` (numeric) and `G` (group), or `K` (Kanji). Natural data formats `C` (attribute control) and Handle are not allowed.<br><br>For a description of Natural data formats, see *Format and Length of User-Defined Variables* and *Special Formats* in the section *User-Defined Variables* in the *Programming Guide*. |
| **Length** | The length of the parameter or `DYNAMIC`.<br><br>This field does not apply to the following Natural data formats: `D` (date), `G` (group), `L` (logical) and `T` (time).<br><br>The Natural data format `A` is restricted to 1073739357 bytes, Natural data format `B` is restricted to 536869678 bytes.<br><br>`DYNAMIC` indicates a dynamic parameter and applies to the Natural data formats `A` and `B`. |
| **Prec** | Only applies to Natural data formats `N` (numeric) and `P` (packed). Optional.<br><br>The precision of the parameter, that is, the number of digits after the decimal point. |

| Field | Description |
|---|---|
| **Dimension 1/2/3** | Only applies to arrays. Optional.<br><br>The first, second and third dimension of the parameter.<br><br>An X-array or an X-group array is specified by entering an asterisk (*) for a dimension.<br><br>See also ***Defining X-Arrays and X-Group Arrays***. |

This section contains information on:

- Defining Groups
- Defining X-Arrays and X-Group Arrays
- Special Considerations for Reliable RPC
- Special Considerations for Calling EntireX RPC Servers

## Defining Groups

You only need to define a group structure for a client Natural object that calls a non-Natural object located on an EntireX RPC server. The group structure must correspond to the IDL definition in EntireX (see ***Special Considerations for Calling EntireX RPC Servers***). A group structure is not required for a client Natural object that calls a subprogram located on a Natural RPC server.

Group arrays and X-group arrays passed from a client Natural object to an interface object must be contiguous. Therefore, we strongly recommend that you always pass a complete array to the object by using asterisk (*) notation for all dimensions. We also strongly recommend that you use identical data definitions in the client Natural program, the interface object and the server program.

See also ***Example 2*** for an example of a group definition.

## Defining X-Arrays and X-Group Arrays

If any dimension of a parameter is extensible, all other dimensions of the parameter are also extensible. If you define extensible and fixed dimensions for a parameter in a subprogram, the **Interface Object Generation** function issues a warning and automatically changes the fixed dimension to an extensible dimension as demonstrated in ***Example 3***. In a group structure, you can define either an extensible or a fixed dimension for each level. There is no automatic change of a fixed dimension to an extensible dimension between levels.

Natural RPC only supports extensible upper bounds. All X-arrays and X-group arrays in the generated `DEFINE DATA PARAMETER` area of the interface object are therefore defined as `(1:*)`.

**Caution:** If you generate an interface object from a subprogram or a PDA that contains an X-array or X-group array with an extensible lower bound, the extensible lower bound will be converted to an extensible upper bound.

For an example of a group with an extensible dimension, see ***Example 3***.

### Special Considerations for Reliable RPC

If you want to use reliable RPC and your parameter definitions do not contain group structures, you have to set `COMPAT IDL` before generating the interface object.

### Special Considerations for Calling EntireX RPC Servers

The attribute definitions on the **Interface Object Generation** screen reflect the perspective of the client. Conversely, the parameter direction in the IDL definition reflects the perspective of the server. This means:

- `OUT` on the **Interface Object Generation** screen corresponds to `IN` in the IDL definition.
- `IN` on the **Interface Object Generation** screen corresponds to `OUT` in the IDL definition.

If you want to call an EntireX RPC server and the parameter definitions on the **Interface Object Generation** screen contain group structures, group structure and attribute definitions on the **Interface Object Generation** screen must correspond to the group structure and parameter direction in the IDL definition.

If you want to call an EntireX RPC server and the corresponding IDL file does not contain group structures, it is recommended to set `COMPAT IDL` before generating the interface object. In this case, the attribute definitions on the **Interface Object Generation** screen must correspond to the parameter direction in the IDL definition.

## Examples of Interface Object Generation

This section provides examples of Natural subprograms and the interface objects generated from them.

The parameter definitions indicated below are extracted from example subprograms, which are supplied in the Natural system library SYSRPC.

### Example 1

The following `DEFINE DATA PARAMETER` area (example subprogram TESTS5) shows four modifiable parameters and the corresponding parameter definitions on the **Interface Object Generation** screen:

```
DEFINE DATA
PARAMETER
  01 #IDENTIFIER  (A10)
  01 #N-OF-ID     (I4)
  01 #FREQ        (P5.2)
  01 #A100        (A100/5,4)
```

| Interface Object Generation | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| | Level | Attr | Type | Length | Prec | Dimension 1 | Dimension 2 | Dimension 3 |
| 1 | 01 | M | A | 10 | | | | |
| 2 | 01 | M | I | 4 | | | | |
| 3 | 01 | M | P | 5 | 2 | | | |
| 4 | 01 | M | A | 100 | | 5 | 4 | |

### Example 2

The following `DEFINE DATA PARAMETER` area (example subprogram TESTS6) shows a nested group structure and the corresponding parameter definitions on the **Interface Object Generation** screen:

```
DEFINE DATA
PARAMETER
 01 GROUP-1(10)
   02 A (A20)
   02 B (A20)
   02 GROUP-2(20)
     03 C (A10/5)
     03 D (A10)
 01 LINE (A) DYNAMIC
```

| Interface Object Generation | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| | Level | Attr | Type | Length | Prec | Dimension 1 | Dimension 2 | Dimension 3 |
| 1 | 01 | M | G | | | 10 | | |
| 2 | 02 | M | A | 20 | | | | |
| 3 | 02 | M | A | 20 | | | | |
| 4 | 02 | M | G | | | 20 | | |
| 5 | 03 | M | A | 10 | | 5 | | |
| 6 | 03 | M | A | 10 | | | | |
| 7 | 01 | M | A | DYNAMIC | | | | |

### Example 3

The following `DEFINE DATA PARAMETER` area (example subprogram TESTS7) shows a nested group structure with extensible dimensions and the corresponding parameter definitions on the **Interface Object Generation** screen.

```
DEFINE DATA
PARAMETER
 01 GROUP-1(10)
   02 A (A20)
   02 B (A20)
   02 GROUP-2(0:*)
     03 C (A10/5)
     03 D (A10)
 01 LINE (A) DYNAMIC
```

| Interface Object Generation | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| | Level | Attr | Type | Length | Prec | Dimension 1 | Dimension 2 | Dimension 3 |
| 1 | 01 | M | G | | | 10 | | |
| 2 | 02 | M | A | 20 | | | | |
| 3 | 02 | M | A | 20 | | | | |
| 4 | 02 | M | G | | | * | | |
| 5 | 03 | M | A | 10 | | 5 | | |
| 6 | 03 | M | A | 10 | | | | |
| 7 | 01 | M | A | DYNAMIC | | | | |

# 113 Generating Multiple Interface Objects


- Using the SYSRPC SGMASS Direct Command ..... 986
- Name Specification and Compression ..... 988

You can generate single or multiple interface objects in either online or batch mode by using the function the direct command `SYSRPC SGMASS`.

You generate interface object from subprograms or parameter data areas (PDAs).

## Using the SYSRPC SGMASS Direct Command

You can enter the `SYSRPC SGMASS` direct command at any Natural command prompt for generating interface objects online.

The section below contains information on:

- Syntax of SYSRPC SGMASS
- SYSRPC SGMASS Report

### Syntax of SYSRPC SGMASS

The syntax that applies to the `SYSRPC SGMASS` direct command is illustrated in the diagram below:

```
SYSRPC SGMASS [name] [compression]
```

The syntactical items *`name`* and *`compression`* are explained in the section *Name Specification and Compression*.

### SYSRPC SGMASS Report

The `SYSRPC SGMASS` direct command produces a report that lists the interface objects generated with the command as shown in the following example:

```
Page       1                                            2006-05-24 16:09:17

    SYSRPC - Interface Object Generation in Library SAGTEST


Generation Criteria:

    Object name or range: RPC*
             Compression: 1


Generation Results:

 Number of objects found:       8
     Maximum send length:  200228
  Maximum receive length: 1024192


Object     Type      Send Length   Receive Length   Message
--------   ----   --------------   --------------   --------------
RPCCALL1   N                 209              202
RPCCALL2   N                 219              240   Compression=2
RPCCALL3   N                 204              193
MORE
```

The report is organized in three sections, which contain the following information:

- **Generation Criteria:**
  The criteria based on which the interface object(s) were generated: a single object name or a range of names (here: `RPC*`) and the compression (here: `1`).
- **Generation Results:**
  The number of objects selected for the interface object generation.

  The maximum buffer sizes all generated interface objects require for sending and receiving data from the client.
- **Object List:**
  The name and type (here: `N` for type subprogram) of each generated interface object. The buffer sizes each object requires for sending (**Send Length**) and receiving (**Receive Length**) data from the client. A possible comment on each generation of an interface object in the **Message** column. In the example above, `Compression=2` indicates that object `RPCCALL2` was not generated with Compression 1 as requested in the command. The object list is sorted in alphabetical order of object names.

If the MORE prompt appears, choose ENTER to scroll to the end of the report.

If the interface object generation fails for single or multiple objects, the report shows the number of objects affected and appropriate error messages.

## Name Specification and Compression

You can specify the objects (subprograms or PDAs) to be selected for interface object generation and the type of compression to be used:

- Name
- Compression

### Name

You can specify an object name or a range of names. The specification of an object name or a range of names is optional.

**Caution:** If you do not specify an object name or a range of names, with few exceptions (see below), all subprograms or PDAs in the current library will be converted to interface objects.

Valid name specifications are described below where *value* is any combination of one or more alphanumeric characters:

| Input | Objects Selected |
|---|---|
| * | All subprograms or PDAs.<br><br>This is the default setting. |
| *value* | A subprogram or a PDA with a name equal to *value*. |
| *value** | All subprograms or PDAs with names that start with *value*. |
| *value*< | All subprograms or PDAs with names less than or equal to *value*. |
| *value*> | All subprograms or PDAs with names greater than or equal to *value*. |

**Exceptions to Names**

In the Natural system library SYSRPC, `SYSRPC SGMASS` exempts from interface object generation all subprograms with names that start with any of the following prefixes: RDS, RPC, NAT, NAD or NSC.

In user libraries, `SYSRPC SGMASS` exempts from interface object generation the subprogram NATCLTGS.

### Compression

You can specify any of the following compression types: `0`, `1`, `2`. The specification of compression is optional. The default type used for interface object generation is `1`.

See also *Using Compression* described in *Operating a Natural RPC Environment* in the *Natural RPC (Remote Procedure Call)* documentation.

# 114 Calculating Size Requirements


- Using the SYSRPC CSMASS Direct Command ..... 992
- Name Specification and Compression ..... 994

The `SYSRPC CSMASS` direct command is used to calculate the buffer size RPC calls without intreface objects require for sending data from the client to the server or vice versa. The indication of the size helps you configure the middleware layer used; for example, the Broker attribute file when EntireX Broker is used.

If desired, you can also perform size calculations for interface objects, even though sizes are already calculated when the interface objects are generated.

`SYSRPC CSMASS` can be used in online or batch mode.

## Using the SYSRPC CSMASS Direct Command

You can enter the command `SYSRPC CSMASS` at any Natural command prompt for calculating size requirements online.

The section below contains information on:

- Syntax of SYSRPC CSMASS
- SYSRPC CSMASS Report

### Syntax of SYSRPC CSMASS

The syntax that applies to the `SYSRPC CSMASS` direct command is illustrated in the diagram below:

```
SYSRPC CSMASS [name] [compression]
```

The syntactical items *`name`* and *`compression`* are explained in the section *Name Specification and Compression*.

### SYSRPC CSMASS Report

The `SYSRPC CSMASS` direct command produces a report that indicates the send and receive length requirements of the subprograms (objects) specified with the command as shown in the following example:

```
Page       1                                            2006-05-24 15:54:12

    SYSRPC - Calculation of Buffer Sizes for RPC Without Interface Objects


Calculation Criteria:

    Object name or range: RPC*
             Compression: 1


Calculation Results:

 Number of objects found:       8
     Maximum send length:  200228
  Maximum receive length: 1024192


Object     Type      Send Length   Receive Length   Message
--------   ----   --------------   --------------   --------------
RPCCALL1   N                 209              202
RPCCALL2   N                 219              240   Compression=2
RPCCALL3   N                 204              193
MORE
```

The report is organized in three sections, which contain the following information:

- **Calculation Criteria:**
  The criteria based on which the calculation was made: a single object name or a range of names (here: `RPC*`) and the compression (here: `1`).
- **Calculation Results:**
  The number of objects selected for the size calculation.

  The maximum buffer sizes all selected objects require for sending and receiving data from the client.
- **Object List:**
  The name and type (here: `N` for type subprogram) of each object selected for the calculation. The buffer sizes each object requires for sending (**Send Length**) and receiving (**Receive Length**) data from the client. A possible comment on each object calculation in the **Message** column. In the example above, `Compression=2` indicates that object `RPCCALL2` was not calculated with Compression 1 as requested in the command. The object list is sorted in alphabetical order of object names.

If the MORE prompt appears, choose ENTER to scroll to the end of the report.

If the size calculation fails for single or multiple objects, the report shows the number of objects affected and appropriate error messages.

## Name Specification and Compression

You can specify the objects (subprograms or PDAs) to be selected for size calculation and the type of compression to be used:

- Name
- Compression

### Name

You can specify an object name or a range of names. If you do not specify a name or a range of names, the size of all subprograms or PDAs contained in the current library will be calculated.

Valid name specifications are described below where *value* is any combination of one or more alphanumeric characters:

| Input | Objects Selected |
|---|---|
| * | All subprograms or PDAs.<br><br>This is the default setting. |
| *value* | A subprogram or a PDA with a name equal to *value*. |
| *value** | All subprograms or PDAs with names that start with *value*. |
| *value*< | All subprograms or PDAs with names less than or equal to *value* |
| *value*> | All subprograms or PDAs with names greater than or equal to *value*. |

### Compression

You can specify any of the following compression types: `0`, `1`, `2`. The specification of compression is optional. The default type used for interface object generation is `1`.

See also *Using Compression* described in *Operating a Natural RPC Environment* in the *Natural RPC (Remote Procedure Call)* documentation.

# 115 Parameter Maintenance


- Invoking Parameter Maintenance ..... 996
- Specifying NTRPC/RPC Keyword Subparameters ..... 996

**Applies to client sessions only.**

The **Parameter Maintenance** function is used to dynamically (within a session) modify keyword subparameters of the `RPC` profile parameter or the `NTRPC` macro.

**Caution:** The parameter modifications are only retained as long as the user session is active; they are lost when the session is terminated.

## Invoking Parameter Maintenance

### To invoke the Parameter Maintenance function

1 In the **Code** field of the **Client Maintenance** menu, enter the following command:

```
PM
```

The **Client Parameter Maintenance** screen appears.

2 Modify the values of the input fields: see *Specifying NTRPC/RPC Keyword Subparameters*.

3 Choose PF3 (Exit) to save any modifications and exit the **Client Parameter Maintenance** screen.

Or:

Choose PF12 (Canc) to exit without saving any parameter modifications.

The **Client Maintenance** menu appears.

## Specifying NTRPC/RPC Keyword Subparameters

In the input fields provided on the **Client Parameter Maintenance** screen, you can modify the settings of the keyword subparameters of the `NTRPC` macro or the `RPC` profile parameter described in the table below:

| Field | Explanation |
|---|---|
| **Timeout** | Specifies the number of seconds the client is to wait for an RPC server response.<br><br>See also the `NTRPC/RPC` keyword subparameter `TIMEOUT` described in the *Parameter Reference* documentation. |
| **Try alternative servers** | Specifies whether an RPC client is to try to execute a service on an alternative server (`ON`) or not (`OFF`). See also *Using an Alternative Server* in the *Natural Remote Procedure Call (RPC)* documentation. |

| Field | Explanation |
|---|---|
| | See also the `NTRPC/RPC` keyword subparameter `TRYALT` described in the *Parameter Reference* documentation. |
| **Compression for auto remote RPC** | Specifies the compression type for an automatically generated RPC call; see *Using Compression* described in the *Natural Remote Procedure Call (RPC)* documentation.<br><br>See also the `NTRPC/RPC` keyword subparameter `COMPR` described in the *Parameter Reference* documentation.<br><br>For more information on automatic RPC execution, see *Working with Automatic Natural RPC Execution* in the *Natural RPC (Remote Procedure Call)* documentation. |

For further information on parameter settings, see *Keyword Subparameters* for *Client and Server*| *Client Only*| *Server Only* in the section *RPC - Remote-Procedure-Call Settings*, in the *Parameter Reference* documentation.

# 116 Server Command Execution


- Using Server Command Execution ..... 1000
- Pinging an RPC Server ..... 1002
- Terminating an RPC Server ..... 1004

The SYSRPC utility provides the server execution commands ping and terminate. They are used to control active servers that have been defined in the service directory. The ping command sends an internal message to the server to verify a server connection. Terminate either sends an internal message to the server requesting termination of a single server task, or issues a command to EntireX Broker requesting termination of all server tasks associated with an EntireX Broker service.

The server execution commands reference the service directory in the library that is defined with the `NTRPC/RPC` keyword subparameter `RPCSDIR` (see the *Parameter Reference* documentation). If `RPCSDIR` is not set (this is the default), the library where you are currently logged on is used. The name of the library is indicated in the upper right corner of the **Server Command Execution screen** shown in the following section.

## Using Server Command Execution

### To use Server Command Execution

1 In the **Code** field of the **Client Maintenance** menu, enter the following command:

```
XC
```

2 Choose ENTER.

The standard view of the **Server Command Execution** screen appears similar to the following example:

```
 15:16:30            ***** NATURAL SYSRPC UTILITY *****              2016-07-18
                          - Server Command Execution -         Library SAGTRPC2

    Cmd Node                          Server                           Message
 1      ETB045
 2   __                               NRPC001                          Natural
 3   __                               NRPC002




Command ===>


Enter-PF1---PF2---PF3---PF4---PF5---PF6---PF7---PF8---PF9---PF10--PF11--PF12---
      Help  ERR   Exit        -H    +H    -P    +P    TOP   BOT   <     Canc
```

The standard view displays the columns **Node**, **Server** and **Message**. The fields under the column **Message** are truncated and display a maximum of 8 characters.

3 If you choose PF11 or enter the less than (<) sign in the Command line at the bottom of the screen, the extended message view of the **Server Command Execution** screen is displayed similar to the following example:

```
 16:36:39            ***** NATURAL SYSRPC UTILITY *****              2016-07-18
                          - Server Command Execution -         Library SAGTRPC2

    Cmd  Server            Message
 1
 2   __  NRPC001           Natural RPC Server 8.3.7 on WNT-x86
 3   __  NRPC002




Command ===>


Enter-PF1---PF2---PF3---PF4---PF5---PF6---PF7---PF8---PF9---PF10--PF11--PF12---
      Help  ERR   Exit        -H    +H    -P    +P    TOP   BOT    >    Canc
```

The extended view allows you to display a maximum of 50 characters of message text in the **Message** column. This view does not display the **Node** column and the fields under the **Server** column are truncated and display a maximum of 16 characters (the standard view shows 30 characters).

If you choose PF11 once more or enter the greater than (>) sign in the Command line, the standard view of the **Server Command Execution** screen is displayed again as shown in *Example of a Standard View*.

This section covers the following topics:

- Line Commands: Server Command Execution

### Line Commands: Server Command Execution

The line commands available on the **Server Command Execution screen** depend on whether they are executed on an EntireX Broker node or an RPC server node. In the following table, an X indicates whether a command is available for a node.

| Line Command | Description | Broker | Server |
|---|---|---|---|
| PI | **Broker** node: Pings all servers defined for the selected EntireX Broker.<br><br>**Server** node: Pings the selected RPC server.<br><br>See also the SYSRPC PING direct command. | X | X |
| TE | Terminates the selected RPC server. | | X |
| TS | Terminates the selected EntireX Broker service. | | X |
| LS | List servers registered on the selected EntireX Broker.<br><br>See also the SRVLIST direct command. | X | X |
| IV | Lists the versions of the selected EntireX Broker and its Command and Information Services (CIS) and the version of the EntireX Broker stub. | X | X |

## Pinging an RPC Server

You can ping an RPC server from the standard or extended message view of the **Server Command Execution** screen or by using the SYSRPC PING **direct command**.

For information on pinging an RPC server by using the Application Programming Interface USR2073N, see the appropriate *Natural RPC (Remote Procedure Call)* documentation.

The following section provides instructions for pinging an RPC server from the standard view of the **Server Command Execution screen**.

### To ping an RPC server from the Server Command Execution screen

1 In the **Cmd** column next to the server(s) to be pinged, enter the following line command:

```
PI
```

as shown in the example of a **Server Command Execution** screen below:

```
 16:41:32            ***** NATURAL SYSRPC UTILITY *****              2016-07-18
                          - Server Command Execution -         Library SAGTRPC2

    Cmd Node                          Server                           Message
 1   __ ETB045
 2   PI                                  NRPC001
 3   PI                                  NRPC002




Command ===>


Enter-PF1---PF2---PF3---PF4---PF5---PF6---PF7---PF8---PF9---PF10--PF11--PF12---
      Help  ERR   Exit        -H    +H    -P    +P    TOP   BOT    >    Canc
```

2 Choose ENTER. The server(s) return the message:

```
Server version on operating system
```

where

*Server* denotes the type of server; *version* denotes the version *operating system* denotes on which operating system the server runs.

Example message:

```
Natural RPC Server 8.3.7.0 on WNT-x86
```

If pinging the server fails and an error occurs instead, you can choose PF2 (ERR) to display RPC-related Natural and EntireX Broker messages as described in *Using the RPCERR Program* (*Monitoring the Status of an RPC Session*, *Natural RPC (Remote Procedure Call)* documentation).

3 To display more of the message text which appears truncated in the standard view of the **Server Command Execution** screen (see also ***Example of a Standard View***) proceed as follows:

Choose PF11.

Or:

In the Command line, enter the less than (<) sign.

This section covers the following topic:

- Using the SYSRPC PING Direct Command

### Using the SYSRPC PING Direct Command

You can ping an RPC server by using the `SYSRPC PING` direct command in online or batch mode.

The following command syntax applies:

```
SYSRPC PING server-name ON broker-name [[PORT] port-number][TRANSPORT {TCP|SSL|NET}]
```

The symbols used in the syntax diagram are explained in the section *Syntax Symbols* in the *Statements* documentation.

The syntax elements are explained in the following table:

| Syntax Element | Format/Length | Description | |
|---|---|---|---|
| *server-name* | A32 | Name of an RPC server or a range of names<br><br>An asterisk (*) selects all names, asterisk notation selects all names that start with the specified value. | |
| *broker-name* | A32 | Name of the EntireX Broker or a range of names<br><br>An asterisk (*) selects all names, asterisk notation selects all names that start with the specified value. | |
| *port-number* | N5 | Port number of the network address used for the server connection.<br><br>Valid values: `0` to `65535` | |
| `TRANSPORT` | A3 | Transport method used by EntireX Broker: | |
| | | `TCP` | TCP/IP protocol |
| | | `SSL` | SSL or TLS (not supported on z/VSE) |
| | | `NET` | Entire Net-Work (not supported on UNIX or Windows) |

## Terminating an RPC Server

The SYSRPC utility provides two commands to terminate a server: `TE` (Terminate RPC Server) and `TS` (Terminate EntireX Broker Service).

`TE` terminates a single RPC server task by sending an internal message to the RPC server. If an RPC server is associated with multiple RPC server tasks (including replicas on mainframe platforms), you can either terminate each RPC server task separately by using `TE`, or terminate all RPC server tasks in one go by using the `TS` command.

`TS` terminates all server tasks associated with an EntireX Broker service by calling EntireX Broker's Command and Information Services (ETBCIS; for details, see the EntireX documentation). The term `service` here summarizes all server tasks that run with the same server name on the same or on different platforms.

You can terminate server tasks from the standard or extended message view of the **Server Command Execution** screen.

The following section provides instructions for terminating a single RPC server task or an EntireX Broker service from the standard view.

For alternative methods of terminating servers, see *Terminating a Natural RPC Server* described in the *Natural RPC (Remote Procedure Call)* documentation.

### To terminate a single RPC server task

1 In the **Code** field of the **Client Maintenance** menu, enter the following command:

```
XC
```

The standard view of the **Server Command Execution** screen is displayed.

2 In the **Cmd** column next to the server(s) to be terminated, enter the following line command:

```
TE
```

(This is similar to entering the command `PI` as shown in the **example** of pinging a server.)

3 Choose ENTER.

The server returns the message:

```
Terminating Server version on operating system
```

where

*`Server`* denotes the type of server; *`version`* denotes the four or five-digit product number; *`operating system`* denotes the operating system the server runs.

Example message:

```
Terminating Natural RPC Server 6.3.1.0 on WNT-x86
```

If terminating the server fails and an error occurs instead, you can choose PF2 (ERR) to display RPC-related Natural and EntireX Broker messages as described in *Using the RPCERR Program* (*Monitoring the Status of an RPC Session*, *Natural RPC (Remote Procedure Call)* documentation).

To display more of the message text which appears truncated in the standard view of the **Server Command Execution** screen:

Choose PF11.

Or:
In the Command line, enter the less than (<) sign.

4 If the **Logon** option is set in the service directory, logon data (user ID, password and library name) is sent to the server with the `TE` command, as is usual for remote `CALLNAT` execution. The **Security Token Data** window pops up and requests input of user ID and password if no Natural Security is installed on the client side and no logon data is set with the Application Programming Interface USR1071N for the current Natural session. See also USR1071N described in *Using Security*, *Using Natural RPC with Natural Security*, in the *Natural RPC (Remote Procedure Call)* documentation.

If `LOGONRQ=ON` (see also *Using Security* in the *Natural RPC (Remote Procedure Call)* documentation) has been set on the server side, logon data must be sent from the client with the `TE` command.

If Natural Security is installed on the server, the logon data transferred must enable a logon to the Natural system library SYSRPC.

### To terminate an EntireX Broker service

1 In the **Code** field of the **Client Maintenance** menu, enter the following command:

```
XC
```

The standard view of the **Server Command Execution** screen is displayed.

2 In the empty column between the sequence number and the **Node** column, in the line which belongs to the server to be terminated, enter the following command:

```
TS
```

(This is similar to entering the command `PI` as shown in the **example** of pinging a server.)

3 Choose ENTER.

The **SYSRPC - Terminating EntireX Broker Service** window appears.

4 If required for the logon, enter the appropriate user ID and password for EntireX Broker.

If you want to terminate server tasks that are involved in a conversation, in the **Terminate immediately** field, enter a `Y` to request immediate termination. If you enter an `N` (this is the default setting), all server tasks involved in a conversation remain operational.

If you do not want this window to appear repeatedly during the current SYSRPC session, choose **Do not show this window again**.

5 Choose ENTER to terminate the EntireX Broker service.

# 117 Listing Servers Registered on EntireX Broker


- Example of an SYSRPC SRVLIST Direct Command ..... 1009
- Viewing a Server List ..... 1009
- Viewing Additional Server Information ..... 1010
- Customizing Server Lists ..... 1011

You can obtain information on RPC servers registered on EntireX Broker by using the `SYSRPC SRVLIST` direct command.

`SYSRPC SRVLIST` sends a call to EntireX Broker requesting information on RPC servers registered on EntireX Broker with the attributes `SERVER-CLASS=RPC` and `SERVICE=CALLNAT`.

You can execute `SYSRPC SRVLIST` online (issued from a Natural command prompt) or in batch mode.

**Note:** When you execute this command online, a window prompts you to logon to the EntireX Broker specified in the command.

The following command syntax applies to `SYSRPC SRVLIST`:

```
SYSRPC SRVLIST server-name ON broker-name [[PORT] port-number][TRANSPORT
{TCP|SSL|NET}][USING{HEAD1MAP|object-name}]
```

The symbols used in the syntax diagram are explained in the section *Syntax Symbols* in the *Statements* documentation.

The syntax elements are explained in the following table:

| Syntax Element | Format/Length | Description | |
|---|---|---|---|
| *server-name* | A32 | Name of an RPC server or a range of names<br><br>An asterisk (*) selects all names, asterisk notation selects all names that start with the specified value. | |
| *broker-name* | A32 | Name of the EntireX Broker or a range of names<br><br>An asterisk (*) selects all names, asterisk notation selects all names that start with the specified value. | |
| *port-number* | N5 | Port number of the network address used for the server connection.<br><br>Valid values: `0` to `65535` | |
| `TRANSPORT` | A3 | Transport method used by EntireX Broker: | |
| | | `TCP` | TCP/IP protocol |
| | | `SSL` | SSL or TLS (not supported on z/VSE) |
| | | `NET` | Entire Net-Work (not supported on UNIX or Windows) |
| *object-name* | A8 | Name of the Natural text object used to customize a server report.<br><br>See also ***Customizing Server Lists***. | |

This section covers the following topics:

## Example of an SYSRPC SRVLIST Direct Command

```
SYSRPC SRVLIST SERV* ON BRK123
```

This command returns data for all servers whose names start with `SERV` on EntireX Broker `BRK123`.

## Viewing a Server List

When you execute the `SYSRPC SRVLIST` direct command online, a **Servers** list screen similar to the example below appears:

```
 13:03:53             ***** NATURAL SYSRPC SRVLIST *****             2016-07-14
                       - Servers registered on BRK123 -
 Cmd  Server           TransRoutine   Requests ConvTimeouts ServersActive Conv>
 ---  ---------------- ------------ ---------- ------------ ------------- ---->
  _   SERVRPC1                               0           60             1
  _   SERVRPC2         SAGTCHA               0           60             1
  _   SERVRPC3                               0           60             1
  _   SERVRPC4         SAGTCHA              76           60             1
  _   SERVRPC5                            2035           60             1
  _   QA42RPC6         RPCTRNS              25          600             2
  _   QA42RPC7                           11190           60             1
  _   QA42RPC8                             206           60             1


Command ===>


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

The columns and column headings on the **Servers** screen are described in the `HEAD1MAP` text object. See also *Customizing Server Lists*.

You can navigate through the list and view additional server information by using the following commands described in the following table.

The leftmost column containing the name of the RPC server is always retained at its position when you scroll right or left in the list.

| Command | Description |
|---|---|
| PF3 | Terminates the command. |
| PF6 | Scrolls data to the leftmost column. |
| PF7 | Scrolls data to the left. |
| PF8 or PF11 | Scrolls data to the right. |
| PF9 | Scrolls data to the rightmost column. |
| I | Line command entered in the **Cmd** column for a listed server.<br><br>Displays additional information on a single RPC server: See also *Viewing Additional Server Information*. |

## Viewing Additional Server Information

You can display additional information on a specific RPC server.

### To display additional information for a single server

- In the **Cmd** column of the **Servers screen**, enter the line command `I` next to the server for which you want to display additional information.

  An **Information on Server** screen similar to the example below appears:

```
15:16:28            ***** NATURAL SYSRPC SRVLIST *****            2016-07-14
                - Information on Server SERVRPC4 on BRK123 -
Description                                                   Value
------------------------------------------------------------ -------------------
Character set used on platform.........................: EBCDIC SNI
Endian type of platform................................: Big endian
Status of user.........................................: Waiting
Kind of conversation for which user waits..............: NEW
Server for which user waits (Class=RPC/Service=CALLNAT)..: SERVRPC4
Number of active conversations of this user............: 0
Number of services active (offered) by this server.......: 1
Elapsed time since the last activity of the user.........: 95
Non-activity timeout in seconds........................: 600
Accumulated time server waited for new conversations.....: 68856
Number of times server waited for new conversations......: 190
Accumulated time server or client waited for messages of>: 0
Number of times server or client waited for messages of >: 0
Sum of conversations for the user since start of session.: 76
Number of UOWs (units of work).........................: 0
IPv4 address of server.................................: 10.20.91.119

Command ===>
```

The screen displays all information EntireX Broker returned for the requested server. See also *Customizing Server Lists*.

## Customizing Server Lists

You can rearrange a **list of servers** or a **server information list** as required by using the Natural `HEAD1MAP` or `HEAD2MAP` text object, respectively. `HEAD1MAP` and `HEAD2MAP` are supplied in the SYSRPC system library.

We recommend that you copy `HEAD1MAP` (list of servers) from the SYSRPC library to a user-defined library before you start editing the list. You can then rename the object and reference it in the `SRVLIST` command. You cannot rename `HEAD2MAP` (server information) list).

The text objects to be used must be contained in the current library, the library specified with the `NTRPC/RPC` keyword subparameter `RPCSDIR` (see the *Parameter Reference* documentation), or the SYSRPC system library if the object name `HEAD1MAP` is used.

`HEAD1MAP` and `HEAD2MAP` contain instructions on how you change a list according to your needs. You can comment out the source code lines for columns and headings not required in your report. You can change code line positions to reorder columns. **Exceptions:**

- For `HEAD1MAP`: You must not comment out or move the first source line containing the `SERVER-NAME` field. You must not change the name of a field in the **Field** column.
- For `HEAD2MAP`: You must not change the name of a field in the **Field** column.

# 118 Overview of SYSRPC Direct and Batch Commands

The following syntax diagram is an overview of SYSRPC direct commands available online and in batch mode.

The comment next to each command indicates the section where the respective command is described in this chapter.

```
SYSRPC [ CSMASS /* Calculating Size Requirements
         PING /* Pinging an RPC Server
         SGMASS /* Generating Multiple Interface Objects
         SM REPLACE /* Replacing Items in the Service Directory
         SRVLIST /* Listing Servers Registered on EntireX Broker ]
```

# XXIII SYSTP Utility

The SYSTP utility is used to monitor and control characteristics of Natural that are specific to TP monitors.

The SYSTP utility provides functions that are available in most environments, under most TP monitors. They are described in ***General SYSTP Functions***.

The SYSTP utility provides additional functions for the following TP monitors:

- CICS
- IMS TM
- TIAM and *open*UTM

These environment-dependent functions are described in the relevant sections of the *SYSTP Utility* documentation.

**Note:** In this documentation, *open*UTM is referred to as UTM.


**Invoking SYSTP and Executing Functions**

**Using SYSTP Utility Screens**

**General SYSTP Functions**

**SYSTP Functions under CICS**

**SYSTP Functions under IMS TM**

**SYSTP Functions under TIAM and UTM**

**SYSTP in Batch for CICS Sessions**

# 119 Invoking SYSTP and Executing Functions

This section provides instructions for invoking the SYSTP utility and executing a SYSTP utility function. You can execute a SYSTP utility function by using either a SYSTP utility menu or a SYSTP direct command.

### To invoke SYSTP and execute a menu function

1 At any command prompt, enter the following system command:

```
SYSTP
```

The **Main Menu** of the SYSTP utility appears.

2 Execute a SYSTP function by entering the character code that corresponds to the function required in the **Code** field and pressing ENTER.

If you enter `E`, the menu for environment-dependent TP-monitor functions appears. This function is not available under Com-plete and TSO.

### To issue a SYSTP direct command

- At any command prompt or in batch mode, use the `SYSTP` command followed by the function code that corresponds to the SYSTP menu function required. You can enter multiple function codes in the sequence in which they are executed from a menu. If you enter multiple function codes, separate them by periods (.).

  For example:

  - `SYSTP  M.A`

    Invokes the **Natural Monitor Menu** (function code **M** in the SYSTP **Main Menu**) and activates the Natural monitor (function code **A** in **Natural Monitor Menu**).

  - `SYSTP S.A.C`

Invokes the **Slot Size Calculation** screen after:
**S** was executed from the SYSTP **Main Menu**,
**A** was executed from the **Natural Swap Main Menu**, and
**C** was executed from the **Natural Swap Administration Menu**.

See also ***SYSTP in Batch for CICS Sessions***.

# 120 Using SYSTP Utility Screens

The functions provided in a SYSTP utility menu can be invoked by either entering a function code or pressing the PF key (if available) that corresponds to the function required.

The line commands and PF keys (or corresponding direct commands) which are available on many of the SYSTP utility screens are described in the following table. You enter a line command in the **C** column next to the required list item on the screen. You enter a direct command in the Command line.

| PF Key or Direct Command | Line Command | Function |
|---|---|---|
| ?<br>or<br>* | ?<br>or<br>* | If entered in the **C** column, all line commands available on the current SYSTP screen are displayed.<br>If entered in the Command line, all direct commands available within the SYSTP utility are displayed. |
| PF1<br>or<br>HELP | - | Provides help information about the SYSTP function currently used. |
| PF3 | . | Exits the current screen and returns to the previous level/screen. |
| PF5 | /<br>or<br>P | Positions the line in which the line command is entered (or where the cursor is placed) to the top of the screen. |
| PF4 | S<br>or<br>U | Displays detailed information about the item contained in the line, in which the line command is entered (or where the cursor is placed). |
| PF6 | - | Scrolls to the beginning of a list. |

| PF Key or Direct Command | Line Command | Function |
|---|---|---|
| or<br>`--` | | |
| PF7<br>or<br>`-` | - | Scrolls one page backward in a list. |
| PF8<br>or<br>+ | - | Scrolls one page forward in a list. |
| PF9<br>or<br>++ | - | Scrolls to the end of a list. |

The headers of all SYSTP statistics screens contain the following information:

- The **User** field with the ID of the current user as assigned by the Natural system variable `*USER` (see the *System Variables* documentation).
- The **TID** field with the terminal ID assigned to the current user by the Natural system variable `*INIT-ID` (see the *System Variables* documentation).

# 121 General SYSTP Functions


- Natural Monitoring (SYSMON) ..... 1022
- Natural Print/Work Files (SYSFILE) ..... 1026
- Natural Swap Information ..... 1027
- Buffer Usage Statistics (BUS) ..... 1031
- Natural Sub-Systems and Roll Server Information ..... 1033
- Natural Thread Usage Statistics ..... 1034
- Natural License Information ..... 1037

# Natural Monitoring (SYSMON)

The Natural monitoring function can be used to view statistics about the Natural programs and terminals used during the current Natural session.

The scope of the Natural monitor is determined by the area in which the statistical data is collected:

- When a global or local monitor buffer pool is used (as assigned by the `BPI` profile parameter or the `NTBPI` macro of the Natural parameter module; see the *Parameter Reference* documentation), program and terminal statistics of all Natural sessions that share this buffer pool are collected.
- When a monitor buffer is used within the thread (as determined by the `MONSIZE` profile parameter; see the *Parameter Reference* documentation), only program statistics and information about the terminal of the current Natural session are collected. For a monitor buffer, we recommend a minimum value of 64 KB.

In addition to defining the scope for the Natural monitor, you have to set the `RDCSIZE` profile parameter, which activates the Natural Data Collector.

**Caution:** When active, the monitoring function can affect overall system performance.

When you invoke the **Natural Monitoring (SYSMON)** function, the **Natural Monitor Menu** appears, which provides the following functions:

- Activate/Deactivate Monitor
- Monitor Status Information
- Display Program/Terminal Statistics

### Activate/Deactivate Monitor

With these functions you can activate or deactivate the monitor function.

When the monitor function is activated, it begins collecting statistical information of current session(s). Once the monitor function is deactivated, a statistical summary is written to the system log file.

### Monitor Status Information

This function provides statistical information about whether monitoring data is collected in a monitor buffer pool and/or the space allocated with the `MONSIZE` parameter, and indicates the size and the address of the allocated space.

### Display Program/Terminal Statistics

You can view statistical information about all Natural programs that have been executed since the monitor was started and the terminals that have been activated since then.

#### To execute the program or terminal statistics function

1 In the **Code** field of the **Natural Monitor Menu**, enter `P` (for program statistics) or `T` (for terminal statistics).

   You can specify selection criteria for the programs/terminal and/or libraries to be viewed: In the **Name of LTERM or Program** field and/or the **Name of Library** field, enter the name of the required item or specify a range of names by using asterisk (*) and/or a wildcard (?). If you leave the fields empty or only enter an asterisk (*), all programs/terminals and libraries are selected.

   Examples of Name Ranges:

   `*CD` selects ABCD, ACD.
   `AB*` selects AB, AB1, ABC, ABEZ.
   `ABC?` selects ABCA, ABCZ.
   `A?C*Z` selects ABCZ, AXCBBBZ and ANCZ.

2 When you have finished specifying selection criteria and press ENTER, a statistics overview screen similar to the example below appears:

```
13:44:35              ***** NATURAL SYSTP UTILITY *****            2008-08-29
 User SAG           - Natural Monitor Program Statistics -       TID TCK8

 C Program  Library       NAT- ADA- Ext- Mean-  Screen I/O   User    Sys  Fetch
   *        *             time time time  time     No   KB    Acc    Acc
 ---------------------------------------------------------------------------
 _ MONMNU1M SYSTP            0    0    0   0.0     18    9      6      0     19
 _ SYSMAP0M SYSTP            0    0    0   0.0      0    0      0      0     52
 _ S2SCOM01 SYSTP            0    0    0   0.0      0    0      0      0     36
 * MONMNU1P SYSTP            0    0    0   0.0      1    0     28      0      6
 _ MONLST1P SYSTP            0    0    0   0.0      3    0     21      0     12
 _ SYSMAP1M SYSTP            0    0    0   0.0     19   13      2      0     20
 _ NAT00059 SYSTP            0    0    0   0.0      0    0      0      0    346
 _ STPMNU1P SYSTP            0    0    0   0.0      4    0     30      4      8
 _ NAT42004 SYSTP            0    0    0   0.0      0    0      0      0      9
 _ STPMMM1M SYSTP            0    0    0   0.0      7    5      4      0      9
 _ SYSTPRET SYSTP            0    0    0   0.0      0    0     13      0      4
 _ NAT00030 SYSTP            0    0    0   0.0      0    0      0      0      4
 _ NSCC0002 SYSTP            0    0    0   0.0      0    0     14      0      7
 _ LOGON    SYSTP            0    0    0   0.0      0    0      0      0      1
   Top of List                                                          1 of 3
Command ===>
Enter-PF1---PF2---PF3---PF4---PF5---PF6---PF7---PF8---PF9---PF10--PF11--PF12---
Cont  Help  Menu  Exit  Sel         --    -     +     ++    Term        Canc
```

This screen lists all programs/terminals and libraries that have been active in your current Natural session.

PF4 invokes a **Selection** window that can be used to specify selection criteria (see Step 1) to reduce the list of items displayed on the screen.

You can use PF10 to toggle between the program statistics and terminal statistics screen.

The columns contained on a statistics overview screen and the corresponding field names on a detailed statistics screen (see column **C** below) are explained in the following table:

| Column | Corresponding Field | Explanation |
|---|---|---|
| **C** | - | Only applies to the statistics overview screen.<br><br>This input field can be used to invoke a detailed statistics screen for a selected program/terminal:<br><br>Next to the list item about which you require more detailed statistics information, enter any character and press ENTER.<br><br>**Note:** If the statistics overview of active programs/terminals is displayed repeatedly, an asterisk (*) appears in the **C** column next to the program/terminal most active since the last repetition. |

| Column | Corresponding Field | Explanation |
|---|---|---|
| **Program** | **Name of program** | Only applies to program statistics.<br><br>Name of the active program. |
| **Terminal** | **Name of terminal** | Only applies to terminal statistics.<br><br>Name (ID) of the active terminal. |
| **Current Program** | **Current program / library** | Only applies to terminal statistics.<br><br>Name of the executed program and the name of the library that contains this program. |
| **Library** | **Name of library** | Only applies to program statistics.<br><br>Name of the library that contains the program that is executed. |
| **NAT-time** | **Time in Natural** | Time in the Natural nucleus and in the interface. |
| **ADA-time** | **Time in Adabas** | Time waiting for response from Adabas. |
| **Ext-time** | **Time in external program** | Time needed by a user-written module. |
| **Mean-time** | **Mean evaluation time** | Elapsed time of one Natural screen transaction. |
| **Screen I/O No** | **Number of Screen I/Os** | Number of screen I/Os. |
| **Screen I/O KB** | **Amount of data transmitted** | Amount of data transferred to or from the screen. |
| - | **Evaluation time > 3 sec** | Only applies to the detailed statistics screen for a terminal.<br><br>Percentage of evaluation time longer than 3 seconds. |
| - | **Evaluation time > 6 sec** | Only applies to the detailed statistics screen for a terminal.<br><br>Percentage of evaluation time longer than 6 seconds. |
| **User Acc** | **Number of user file accesses** | Counter for accesses to Adabas user files. |
| **Sys Acc** | **Number of system file accesses** | Counter for accesses to Natural system file, including fetches. |
| **Fetch** | **Number of fetches** | Counter for total number of fetches. |

## Natural Print/Work Files (SYSFILE)

This function provides information about available work files and print files.

You can also invoke this function with the system command `SYSFILE` described in the *System Commands* documentation.

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

This function can also be used in batch mode for CICS sessions.

When you invoke this function, the **Work File Information** screen appears with a list of all work and print files defined. The following information is provided for each file:

| Column | Explanation |
|---|---|
| **No.** | Number of the work/print file. |
| **Type** | Type of assignment; that is, the operating system, TP monitor or Natural product file to which the work/print file is assigned. |
| **Name** | Name of the work/print file. |
| **Recfm** | Record format of the work/print file. |
| **Lrecl** | Logical record length of the work/print file (if applicable). |
| **Blksz** | Block size of the work/print file. |
| **Status** | One of the following statuses:<br>`Available for Input/Output`<br>or<br>`Open for Input/Output` |

Under z/VSE, the logical-unit assignments are also displayed.

### Commands for Natural Print/Work Files

In addition to the commands described in *Using SYSTP Utility Screens*, the **Work File Information** screen provides the following PF keys and line command:

| PF Key | Line Command | Function |
|---|---|---|
| PF10 | - | Scrolls to the list of print files. |
| PF11 | - | Scrolls to the list of work files. |
| - | D | Displays the corresponding Natural control block (work file area) in dump format (for internal use by Software AG technical support). |

## Natural Swap Information

This function is only available under CICS and UTM.

The swap pool manager enables online monitoring and control of the Natural swap pool. This section describes how the swap pool manager is used rather than how the swap pool operates. For further information about the operation of the Natural swap pool, see *Natural Swap Pool* in the *Operations* documentation.

When you invoke this function, the **Natural Swap Main Menu** appears, which provides the following functions:

- Administration
- Debugging Facilities
- Information
- Maintenance Services
- Status Information

### Administration

- Slot Size Calculation
- Change Swap Pool Status

- Update Reorg Control Data

#### Slot Size Calculation

This function displays the optimum values for the layout of the swap pool based on the current usage.

You can store these values to be used for a later initialization/reorganization (once they have been stored, they can also be maintained with the **Maintenance Services** function).

You can also initiate a swap pool reorganization using these values.

For further information, see the online help about this function.

#### Change Swap Pool Status

This function is used to activate or deactivate the Natural swap pool. In addition, you can modify the wait time and the number of waits for swap pool synchronization.

For further information, see the online help about this function.

#### Update Reorg Control Data

With this reorganization function, you can modify the most important parameters in swap pool management. You must enter a valid password to modify the values.

For further information, see the online help about this function.

### Debugging Facilities

This function is only available under UTM.

**Caution:** Do not use this function without prior consultation of Software AG technical support.

With this function, it is possible to activate or deactivate an internal screen debugging buffer. Activation of the screen debugging buffer is used to locate terminal I/O inconsistencies if they occur. The function records information about the last three terminal I/O sequences. The buffer has a size of 3 KB and is used in a wrap-around procedure.

In addition, you can activate/deactivate a trace function for asynchronous write operations to the Natural roll file.

For further information, see the online help about this function.

## Information

- Show Addresses
- Show Summary of Buffer Usage
- Show Swap Pool Information
- Show Logical Swap Pools
- Show Reorg Control Data
- Show Swap Pool Usage
- Create Statistics List

### Show Addresses

This function displays the addresses of various pools.

### Show Summary of Buffer Usage

This function is used to optimize the sizes of the various Natural buffers and the Natural user threads (see `MAXSIZE` in the *Operations* documentation). It activates, deactivates and displays a summary of Natural buffer usage.

The activation and deactivation of buffer statistics can only be performed with a valid password. For the display of buffer statistics, no password is required.

The buffers displayed are the same as those displayed by the function **Buffer Usage Statistics (BUS)**.

### Show Swap Pool Information

This function displays information about the swap pool currently in use, including control/statistics data and memory sizes.

The individual items of information shown are explained in the online help about this function.

### Show Logical Swap Pools

This function displays the current table of logical swap pools.

On the table, you can mark a specific logical swap pool with any character to get additional information about it.

The individual items of information shown are explained in the online help about this function.

#### Show Reorg Control Data

This function displays all information related to the swap pool reorganization.

The swap pool reorganization table is displayed in the left half of the screen. The table contains cumulative statistics about the comparative sizes between compressed Natural user threads and standard slot size. The table is cleared with each reorganization of the swap pool. The left half of the table shows how often and to what extent the user threads are larger than the standard slot size. The right half of the table shows how often and to what extent the user threads are smaller than the standard slot size. Sizes in this half of the table are expressed in units that are dependent on the factor specified by the swap pool manager.

In the row labeled **n**, count is taken of user threads which exceed/fall short of the standard slot size by over 9 pages/units. The average length of these user threads is displayed in the row labeled **Av.+n**.

The individual items of information shown are explained in the online help about this function.

#### Show Swap Pool Usage

This function displays information about the usage of the swap pool since its initialization or the last reorganization.

The individual items of information shown are explained in the online help about this function.

#### Create Statistics List

This function is used to create a list of the current swap pool usage statistics:

- Under UTM, the swap pool statistics are written to SYSLIST.
- Under CICS, the swap pool statistics are written to the CICS destination specified with the parameter `MSGDEST` of the macro `NCMPRM` or `NTCICSP` (depending on the Natural CICS Interface version installed) described in the *TP Monitor Interfaces* or *Parameter Reference* documentation, respectively.

### Maintenance Services

- Parameter Maintenance

- Password Maintenance

### Parameter Maintenance

This function is used to change online the parameters for the initialization or reorganization of the swap pool.

The subfunctions as well as the individual items that can be modified are explained in the online help about this function.

The use of this function is password-protected (see below).

### Password Maintenance

This function is used to change or recover the password used for the **Parameter Maintenance** function.

The initial password is `SYSTP`.

### Status Information

With this function, you can display the current status of the Natural swap pool, of the summary of buffer usage and of the UTM screen debugging.

## Buffer Usage Statistics (BUS)

This function provides statistical information on the usage of Natural buffers: which buffers are allocated for the current Natural session, and how much buffer space is being used. In addition, the **Total** figures at the end of the list allow you to draw conclusions about the efficiency of buffer compression.

This section covers the following topics:

- Invoking BUS
- Information provided by the BUS Screen

- BUS Commands

## Invoking BUS

You can invoke this function either from the **SYSTP** menu or with the system command `BUS`. See also ***Invoking SYSTP and Executing Functions***. As a result, a list is displayed showing all buffers that are actually being used in the current Natural session.

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

## Information provided by the BUS Screen

This section gives an overview on items displayed on **Buffer Usage Statistics** screen:

| Column | Explanation |
|---|---|
| **No.** | Buffers are numbered sequentially in order of allocation. |
| **Name** | Name of the buffer. Only those buffers which have actually been requested in the current session are listed. |
| **Type** | Type of the buffer:<br>`blank` Fixed/static Natural buffer.<br>`V` Variable buffer allocated in the Natural thread.<br>`O` Variable buffer currently allocated outside the Natural thread. The buffer is copied and compressed into the thread at the next terminal I/O.<br>`P` Physical buffer allocated in a Natural work pool (see also the `WPSIZE` profile parameter described in the *Parameter Reference* documentation). The buffer is released at the next terminal I/O. |
| **Size** | Size of the buffer (in bytes). |
| **Used** | Number of bytes currently being used. This value is used for buffer compression in environments using threads (for example, CICS or UTM). |
| **Perc.** (**Used**) | Percentage currently being used; that is, the value of the **Used** column in relation to the value of the **Size** column. |
| **MaxUsed** | Maximum number of bytes which have been used in the course of the current session so far (*not* the size being used at present). |
| **Perc.** (**MaxUsed**) | Percentage of current session usage; that is, the value of the **MaxUsed** column in relation to the value of the **Size** column. |

| Column | Explanation |
|---|---|
| **MaxSize** | Maximum size (in bytes) that has been allocated to the buffer in the course of the current session so far (applies to variable buffers only). |
| **Perc.** (**MaxSize**) | Maximum size allocated so far (value of the **MaxSize** column) in relation to the current size (value of the **Size** column; applies to variable buffers only). A percentage of 10000 or more is indicated by `9999.9` displayed intensified. |
| **At the end of the list, the following information is displayed:** | |
| **ThrdSize** | Current size (in KB) of the Natural thread. |
| **Total** | Sums of all buffer sizes (in both bytes and KB) and percentages used/allocated. These totals can also be displayed with PF10 (see below).<br><br>For **MaxSize**, the total shows the maximum additional amount of thread size that would have been needed in the course of the session so far. |

### BUS Commands

In addition to the commands described in *Using SYSTP Utility Screens*, the **Buffer Usage Statistics** screen provides the following PF keys and line command:

| PF Key | Line Command | Function |
|---|---|---|
| PF4 | `D` | Displays the contents of the buffer marked with the cursor/command in dump format (for internal use by Software AG support). |
| - | `S` | Displays details of the selected buffer in a tabular overview, for example **Thread Offset** , **Address** or **Buffer Size**. |
| PF10 | - | Displays the **Total** buffer usage figures. |
| PF11 | - | Displays the relative addresses of the buffers, that is, relative to the input/output control buffer (IOCB). |
| PF12 | - | Displays the buffer compression optimization degree for all buffers in a separate column **CmprOpt**.<br><br>**Note:** By pressing PF12 twice the table is reverted to the original state. |

## Natural Sub-Systems and Roll Server Information

This function displays a list of the Natural subsystems available in your environment and the current status (active or inactive) of authorized service managers and roll servers associated with the subsystems.

In addition to the commands described in *Using SYSTP Utility Screens*, the **Natural Sub-Systems** screen provides the following line commands:

| Line Command | Function | Operating System |
|---|---|---|
| B | Displays information about the buffer pool and buffer pool cache (if available) such as name, type and size. | z/OS and z/VSE |
| D | Displays server directory entries from the Session Information Pool (SIP) in dump format (for internal use by Software AG technical support).<br><br>For information about SIP, refer to *Authorized Services Manager under z/OS* in the *Operations* documentation. | z/OS |
| L | Displays and resets entries in the roll server file directory. | z/OS |
| R | Displays roll server statistics, such as the number of roll server slots and roll server files, roll-out and roll-in activities, as well as roll file I/O.<br><br>This information can help tune the roll server as described in *Roll Server Performance Tuning* in the *Operations* documentation. For example, you can use the statistics data to determine an optimum thread size or roll file size for a Natural application. | z/OS |
| S | Displays a list of Zaps applied to the Authorized Services Manager. | z/OS |
| U | Displays server directory entries from the Session Information Pool (SIP).<br><br>For information about SIP, refer to *Authorized Services Manager under z/OS* in the *Operations* documentation. | z/OS |
| Z | Displays a list of Zaps applied to the roll server. | z/OS |

## Natural Thread Usage Statistics

This function is only available under CICS, Com-plete, IMS TM and UTM. It is not available in a z/OS Parallel Sysplex environment.

This function allows you to determine an optimum thread size or roll file size for a Natural application.

You should activate this function only when needed, and deactivate it after you have determined your optimum thread size, because this function occupies space in the Natural buffer pool. When you deactivate it, the space in the buffer pool becomes available again.

### To execute the Natural Thread Usage Statistics function

1 Define an oversized thread in the range of 512 to 1024 KB for your Natural application. Take into account the number of Natural add-on products used.

2 Start your Natural application, either in production or in test mode.

3 Invoke the SYSTP utility and choose the **Natural Thread Usage Statistics** function by entering T in the **Code** field of the SYSTP **Main Menu**.

The **Thread Usage Statistics** screen appears.

The columns contained on this screen are described in the following table.

4 Enter `A` in the **Code** field to activate the function.

A message appears indicating that the **Thread Usage Statistics** function has been activated.

5 Use your Natural application under typical production conditions.

The **Thread Usage Statistics** function runs in the background and logs the buffer sizes used.

6 When you want to view the statistics, again, invoke the **Thread Usage Statistics** screen and enter one of the following in the **Code** field:

```
S
```

to show the statistics, or

```
P
```

to print the statistics, or

```
G
```

to show `GETMAIN` statistics (see *Show Physical GETMAIN Statistics*), or

```
D
```

to print the statistics and deactivate the function.

We recommend that use function code `D` after function completion to free buffer pool space.

The following information is displayed on the **Thread Usage Statistics** screen:

| Column | Explanation |
|---|---|
| **No.** | Buffers are numbered sequentially in order of allocation. |
| **Ext. Buffer** | Sizes of these buffers are defined externally (in the Natural parameter module). |
| **Defined Size** | Buffer size as defined in the Natural parameter module. |
| **Max. Allocated Size** | Maximum buffer size allocated.<br><br>Note that for the internal BB area, 14368 bytes are added to the value of the `ESIZE` profile parameter (see the *Parameter Reference* documentation). |
| **Max. Used Size** | Maximum buffer size used. |
| **Sum of external buffer sizes** | Total of all buffer sizes defined in the Natural parameter module. |
| **Sum of internal buffer sizes** | Total of all buffer sizes requested by Natural internally. |

| Column | Explanation |
|---|---|
| **Sum of physical GETMAINs** | Total of all physical GETMAINs for the Natural work pools and the variable Natural buffers outside the Natural user threads. |
| **Max. used thread length** | Maximum thread length used by Natural. Define this length as your minimum (optimum) Natural thread length.<br><br>Round it up to the next KB number that can be divided by 2. |
| **Max. compressed thread length** | Maximum length of a compressed Natural thread that was written to the Natural roll file.<br><br>Define this length as your minimum (optimum) Natural roll file length. |

### Show Physical GETMAIN Statistics

The physical GETMAIN statistics provide information about all physical GETMAINs relevant for the Natural work pools and the variable Natural buffers outside the Natural user threads. They indicate the original buffer sizes (**Org. Size**) during the startup of a Natural session, the number of physical GETMAINs, (**No.**) the buffer length for the physical GETMAIN (**Bytes**) and the buffer position (**P**), above or below the 16-MB line.

The statistics data always refers to the buffers with the greatest lengths requested within a terminal I/O, for all users of the Natural application. The statistics provides a maximum of six entries for each buffer. These entries can be overwritten through the wrap-around procedure. The highest number equals the maximum number of the physical GETMAINs within a terminal I/O, for each buffer concerned.

The first two entries in the statistics refer to the Natural work pools (if available) above (**WRK-POOLA**), respectively, below (**WRKPOOLB**) the 16-MB line. Here, the highest physical GETMAIN number refers to the amount of work pools simultaneously available during the terminal I/O. The sum of all work pool lengths amounts to the total storage requirement of the work pools within a terminal I/O.

All subsequent statistics entries refer to the physical GETMAINs for the variable Natural buffers, which either could not be defined in the Natural user thread due to insufficient space, or were increased outside the Natural user threads. For these buffers, the highest physical GETMAIN number indicates the greatest space requirement for each buffer within a terminal I/O. The total storage space requested earlier was freed before each of the following physical GETMAINs. That is, the sum of all physical GETMAINs with the highest number shows the maximum storage requirement for the variable buffers outside the Natural user threads during a terminal I/O, for all users of the Natural application.

## Natural License Information

This function invokes a screen that displays the contents of the license file. You can choose between the following formats:

| Character Code | Format |
|---|---|
| L | XML, left justified (default) |
| S | Structured XML |
| F | Plain text |

**Notes:**

1. To toggle between formats, enter PF5.
2. You can invoke this function either from the **SYSTP** menu or with the system command `SYSTP` followed by function code `L` and optionally a character code (`S` or `F`), for example, `SYSTP L.F` or `SYSTP L F`. See also *Invoking SYSTP and Executing Functions*.
3. In a z/OS environment, you can press PF10 (zIIP) to toggle between the displays of the product license files for Natural and Natural zIIP Enabler.

# 122 SYSTP Functions under CICS


- Natural User Sessions ..... 1040
- Natural Roll Facilities ..... 1045
- Natural Thread Groups ..... 1046
- Natural Storage Threads ..... 1047
- NCI Global System Information ..... 1048
- NCI Generation Options ..... 1050
- Natural Thread Group Definitions ..... 1051
- Own Natural User Session ..... 1052
- CICS Task Information ..... 1052
- System Administration Facilities ..... 1052

The SYSTP utility provides functions that are specific to CICS.

**To invoke specific SYSTP functions under CICS**

- In the **Code** field of the SYSTP **Main Menu**, enter E for **Environment-Dependent Functions**.

  From the menu displayed then, you can select the functions explained in this section.

**Note:** In the remainder of this section, the Natural CICS Interface is also referred to as NCI.

## Natural User Sessions

This function is used to display a list of active user sessions in a Natural environment.

When you invoke this function, the **Natural User Sessions** screen appears, which displays the following information:

| Column | Explanation |
|---|---|
| **Term ID** | Unique terminal ID within CICS associated with the Natural session. |
| **User ID** | Natural user ID of the Natural session. |
| **Tran** | CICS transaction ID under which Natural session is currently running. For pseudo-conversational sessions, this is the pseudo-conversational restart transaction ID. |
| **Start Date / Time** | Starting date and time of the Natural session. |
| **Last Act** | Time of last screen output. |
| **Stat** | Session status: see *Operational Status*. |
| **Program** | Natural program currently active. |
| **Library** | Natural library in which the user is currently working. |

If you press PF10, the display of the session date and time is replaced by the following session resource data:

| Column | Explanation |
|---|---|
| **Thrd Grp** | Thread group to which user is assigned. |
| **Thread** | Name of thread last used. |
| **Roll Fac** | Assigned roll facility. |

This section covers the following topics:

- Commands for Natural User Sessions

- Natural User Session Statistics

## Commands for Natural User Sessions

In addition to the commands described in *Using SYSTP Utility Screens*, for each item displayed on the **Natural User Sessions** screen, you can execute one of the following line commands.

| Line Command | Function |
|---|---|
| C | Cancel session.<br><br>Invokes a confirmation window where you can enter YES to mark a session for termination. The session selected is then flagged with number (#) signs that appear in the column **User ID**. Additionally, for the session concerned, the operational status Purged by Admin appears on the **Natural User Session Statistics** screen of the user (administrator) who executed the cancel command.<br><br>The session actually terminates when the owner of the session marked for termination performs the next terminal I/O, Adabas call or external program call. The session owner then receives a corresponding termination notification. |
| F | Flush session.<br><br>Invokes a confirmation window where you can enter YES to terminate a session immediately. The session terminated is then flagged with number (#) signs that appear in the column **User ID**. The user (administrator) who terminated the session receives a termination message when trying to invoke the **Natural User Session Statistics** screen for the session terminated but still listed on the **Natural User Sessions** screen. The session owner receives a corresponding termination notification. |
| R | Reactivate session.<br><br>Reverses a C (Cancel) command as described earlier. The R command removes the termination flags set for a session and resets the session status to active. Note that you cannot reactivate a session that has been terminated with the F (Flush) command described above. |
| W | Wake up session.<br><br>Reactivates immediately a session that has been suspended by a CMROLL call with a non-zero wait interval specified with the MAXROLL profile parameter (see the *Parameter Reference* documentation). See also the example program SUSPEND supplied in the Natural system library SYSEXTP. |

### To reduce the number of list items by specifying selection criteria

- On the **Natural User Sessions** screen, press PF4.

  The **Selection for User Sessions** window appears where you can select user sessions by inactivity date and time, inactivity time interval, Natural server ID, CICS system ID, terminal ID, user ID or transaction ID.

Inactivity date and time and inactivity interval list all sessions that were not active before the date and time specified or before the time interval specified.

A date must be specified in the format *YYYY-MM-DD* (*YYYY* = year, *MM* = month, *DD* = day). The time must be specified in the format *HH*:*II*:*SS* (*HH* = hours, *II* = minutes, *SS* = seconds).

To specify a range of IDs, use the asterisk (*) as a leading or trailing character or use the question mark (?) as a wildcard character. See also ***Examples of Name Ranges***.

### Natural User Session Statistics

For each session displayed on the **Natural User Sessions** screen shown above, additional information can be displayed by invoking the **Natural User Session Statistics** screen with the `S` or `U` line command. The following section describes the information provided on this screen.

All sizes on the **Natural User Session Statistics** screen are in KB unless otherwise indicated in the field descriptions below.

| Field | Explanation |
|---|---|
| **Started** | Day, date and time when the session was started. |
| **Last Actions** | Date and time when the user was active last. |
| **User** | Natural user ID as assigned by the Natural system variable `*USER` (see the *System Variables* documentation). |
| **at Terminal** | ID of the terminal associated with the Natural session as assigned by the Natural system variable `*INIT-ID` (see the *System Variables* documentation). |
| **Transid** | Pseudo-conversational transaction ID under which Natural is running. |
| **Task # in** | Task number assigned by CICS followed by the ID of the CICS region. |
| **Cur Strg Used** | Current amount of storage used by this session. |
| **Max Strg Used** | Maximum amount of storage ever used by this session. |
| **Thread Size** | Size of this thread. |
| **Thread Name** | Name of the thread used last. For threads allocated by using `GETMAIN`, the thread name is composed of the prefix `NSCP` followed by the terminal ID. |
| **Thread Group** | Name of the associated thread group (triggered by starting the transaction ID). |
| **of Type** | Type of thread used for thread group:<br>`SHR` — Permanent storage threads.<br>`GETM` — Storage threads allocated by using `GETMAIN`.<br>`NONE` — No threads used; all Natural storage requests are passed to CICS. |

| Field | Explanation |
|---|---|
| **Natural Library** | Natural library ID as assigned by the Natural system variable `*LIBRARY-ID` (see the *System Variables* documentation). |
| **Natural Program** | Name of the Natural program currently used by the session as assigned by the Natural system variable `*PROGRAM` (see the *System Variables* documentation). |
| **Line No.** | Line number in the Natural program currently used by the session. |
| **Operational Status** | See ***Operational Status*** in the following section. |
| **Roll Facility** | Name of associated roll facility. |
| **Roll Recs (Last)** | Number of records written to roll facility for last roll-out. |
| **Roll Recs (Max)** | Maximum number of records ever written during roll-out. |
| **Roll Record Size** | Record size of this roll facility. |
| **Slot Size** | Number of records required to roll-out a thread completely. |
| **Restart Rec. No.** | Number of the record that contains roll-out control information; this record must be rolled in first.<br><br>**VSAM Roll Files:**<br><br>The following applies to VSAM roll files only.<br><br>The relationship between restart record number (RecNum), slot number (SN) and slot size (SZ) is:<br><br>RecNum = (SN-1) * SZ + 2<br><br>or<br><br>SN = (RecNum-2) / SZ + 1 |
| **Slot Number** | Number of slot in VSAM roll file belonging to this session (for VSAM only). See also ***VSAM Roll Files*** in **Restart Rec. No.** above. |
| **Compressed Length** | Amount of relevant storage currently swapped/rolled out. |
| **Session Resumes** | Total number of session resumes. |
| **Swap-Ins** | Number of session resumes with swapping in from swap pool. |
| **Thread Switches** | Number of session resumes with swapping/rolling into a thread which is different to the one the session had been in before. |
| **Roll-Ins** | Number of session resumes with rolling in from roll facility. |
| **Region Switches** | Number of CICS region switches. |
| **OpSys Switches** | Number of operating system image switches in a z/OS Parallel Sysplex environment. |

### Operational Status

This field indicates any of the following operational statuses:

| Status | Abbreviation | Description |
|---|---|---|
| Active | Act | Currently active. |
| Inactive | Ina | Inactive, still in thread. |
| Swapped | Swp | Swapped, in swap pool. |
| Rolled out | Rld | Rolled out, in roll facility. |
| Wait (Init) | WtI | Waiting for thread on session initialization. |
| Wait (Resume) | WtR | Waiting for thread on session resume. |
| Initializing | Int | Initializing session. |
| Resuming | Res | Resuming session, in thread, not active yet. |
| Suspending | Sus | Suspending session. |
| Terminating | Trm | Terminating session. |
| Swapping out | Swo | Session swapping out. |
| Swapping in | Swi | Session swapping in. |
| Rolling out | Out | Rolling out from thread or swap pool. |
| Rolling in | In | Rolling in from roll facility. |

The following additional information can appear in **Operational Status**:

| Status | Description |
|---|---|
| Conversational | Dialog-oriented session (PSEUDO=OFF) as opposed to pseudo-conversational/transaction-oriented session.<br><br>See also the PSEUDO profile parameter described in the *Parameter Reference* documentation. |
| Forced Conversational | Last screen I/O of a PSEUDO=ON session was conversational.<br><br>See also the PSEUDO profile parameter described in the *Parameter Reference* documentation. |
| No-Roll | Session is not allowed to roll. |
| Compressed | Session is compressed (in swap pool or roll facility). |
| Thread Switched | Thread currently used is not the same as used before. |
| Thread Locked | Session kept from switching threads (for example, RELO=OFF); can also force No-Roll/Conversational status.<br><br>See also the RELO profile parameter described in the *Parameter Reference* documentation. |
| Purged by Admin | Session canceled by administrator (flag set). |
| Spool Task | Task is a spool/print task. |
| Asynchronous Task | Task is an asynchronous task, not bound to a terminal. |

## Natural Roll Facilities

This function is used to display which swap files are available for rolling out user work areas to make room in the swap pool for active users. These swap files are known as roll facilities.

When you invoke this function, the **Natural Roll Facilities** screen appears for the current CICS region (as indicated by the CICS ID in the screen title). For each roll facility, the following information is displayed:

| Column | Explanation |
|---|---|
| **Facility Name** | `TEMPSTOR` is used for auxiliary temporary storage, `MAINSTOR` for main temporary storage, and remaining file names are VSAM roll files as defined in the CICS file control table (FCT). `none` denotes that no roll facility is used. |
| **Record Size** | Record size of this roll facility. |
| **Slot Size** | This column is displayed by defaultNumber of records required to roll out a thread completely (maximum thread size divided by record size, rounded up). |
| **No. of Slots** | Number of sessions which fit into this roll file (number of file records divided by slot size, rounded down); applies to VSAM roll files only. |
| **Facility Users Cur / Max** | Current (**Cur**) and maximum (**Max**) number of user sessions assigned to this roll facility. |
| **Roll Counts Out / In** | Number of session roll operations from or into this roll facility. |
| **CI Size*** | Control Interval size of the VSAM roll file. |
| **No. of Records*** | Number of records in the VSAM roll file |
| **Roll File Initialization* Date and Time** | Date and time when the VSAM roll file was formatted by the NCISCPRI utility program. |
| **Status** | Indicates `Full` if the facility users equal the number of available slots. |

* These columns are only displayed if you press PF10. If pressed again, it switches back to the default display.

In addition to the commands described in *Using SYSTP Utility Screens*, the **Natural Roll Facilities** screen provides the following PF keys and line commands:

| PF Key | Line Command | Function |
|---|---|---|
| PF4 | `U` | Invokes the **Natural User Sessions** screen (see the relevant section) for the marked roll facility which displays all Natural user sessions that use this facility. |
| PF10 | - | Toggles between the display of the following columns:<br><br>**Slot Size**, **No. of Slots**, **Facility Users** and **Roll Counts**<br>(default display) |

| PF Key | Line Command | Function |
|---|---|---|
| | | and<br><br>**CI Size**, **No. of Records** and **Roll File Initialization**. |

## Natural Thread Groups

This function is used to display which thread groups are available to Natural.

When you invoke this function, the **Natural Thread Groups** screen appears for the current CICS region (as indicated by the CICS ID in the screen title). For each thread group, the following information is displayed on this screen:

| Column | Explanation |
|---|---|
| **Group Name** | Thread group name. |
| **Group Users Cur / Max** | Current (**Cur**) and maximum (**Max**) number of users assigned to this thread group. |
| **Thread Type** | Type of thread used: see **Natural User Session Statistics**. |
| **Thread Size** | Thread group's common thread size. |
| **Strg Used** | Maximum storage ever used by a session that uses this thread group. |
| **TCBs** | Maximum number of sessions concurrently active. |
| **Queue Sizes Cur / Max / AtMax** | Current (**Cur**) and maximum (**Max**) queue size for the thread group's central wait queue and the number of times the maximum was reached (**AtMax**).<br><br>Only applies if the parameter THREADS has been defined as greater than zero for this thread group. See also *THREADS - Number of Threads or Tasks Per Thread Group* in the *TP Monitor Interfaces* documentation. |
| **Max Compr*** | Maximum compressed length of this thread group. |
| **Active Tasks* Cur / Max / AtMax** | Current and maximum number of CICS tasks such as sessions concurrently active in a thread group and the number of times this maximum was reached. |
| **VSAM Aux / Main** | Roll facilities defined for group; CICS temporary storage (auxiliary or main) always backs up VSAM if VSAM roll files are not available or full. |

* These columns are only displayed if you press PF10. If pressed again, it switches back to the default display.

### Commands for Natural Thread Groups

In addition to the commands described in *Using SYSTP Utility Screens*, the **Natural Thread Groups** screen provides the following PF keys and line commands:

| PF Key | Line Command | Function |
|---|---|---|
| PF4 | U | Displays all **Natural user sessions** (see the relevant section) that use the thread group marked with the cursor/command. |
| PF10 | - | Toggles between the display of the following columns:<br><br>**TCBs** and **Queue Sizes**<br>(default display)<br><br>and<br><br>**Max Compr** and **Active Tasks**. |
| - | T | Displays **Natural storage threads** (see below) for the thread group marked with the cursor/command. |
| - | D | Displays **Natural thread group definitions** (see the relevant section) for the thread group marked with the cursor/command. |

## Natural Storage Threads

This function is used to display information about the storage threads in the Natural environment.

When you invoke this function, the **Natural Storage Threads** screen appears for the current CICS region (as indicated by the CICS ID in the screen title). The screen displays the following information:

| Column | Explanation |
|---|---|
| **Thread Name** | Name of the thread. |
| **Grp No.** | Number of the group to which this thread belongs. |
| **Thrd Size** | Usable thread size. |
| **Strg Used** | Maximum storage ever used by a session that uses this thread. |
| **Use Count** | Number of times this thread has been selected for processing. |
| **Roll-Ins Log. / Phys.** | Number of roll-in operations:<br><br>**Log.** — Session resumes.<br><br>**Phys.** — Roll-in from roll facility. |

| Column | Explanation | |
|---|---|---|
| **Queue Sizes Cur / Max / AtMax** | Number of users waiting in the queue: | |
| | **Cur** | Current number of users queuing on thread. If this number *n* is greater than 1, *n* minus 1 users are waiting. |
| | **Max** | Maximum queue count for this thread. |
| | **AtMax** | Number of times at maximum. |
| **Term ID** | Terminal ID belonging to the Natural session whose data are in this thread. | |
| **Task No.** | ID of CICS task currently active in this thread. If no ID is displayed, no session is active is this thread. | |

### Commands for Natural Storage Threads

In addition to the commands described in ***Using SYSTP Utility Screens***, the **Natural Storage Threads** screen provides the following line commands and PF key:

| Line Command | PF Key | Function |
|---|---|---|
| C | - | See ***Cancel session*** in *Natural User Sessions*. |
| D | PF11 | Displays **Natural thread group definitions** (see the relevant section). |
| F | - | See ***Flush session*** in *Natural User Sessions*. |
| G | PF10 | Displays **Natural thread group** (see the relevant section). |
| R | - | See ***Reactivate session*** in *Natural User Sessions*. |
| U | - | Displays statistics about the Natural user sessions currently active in the thread. See also ***Natural User Session Statistics***. |

## NCI Global System Information

This function is used to display data on the system directory.

When you invoke this function, the **Global System Information** screen appears for the current CICS region (as indicated by the CICS ID in the screen title). The screen provides the following information:

| Field | Explanation |
|---|---|
| **Natural User Sessions** | Current (**Cur**) and maximum (**Max**) number of Natural sessions in the system. |
| **Concurrent SCP Active** | Current (**Cur**) and maximum (**Max**) number of concurrent system control program (SCP) requests.<br><br>SCP requests are: session initialization, session suspension, session resumption and session termination. |
| **SIR Block Extensions** | Current (**Cur**) and maximum (**Max**) number of local SIR block extensions. |
| **Slots in 1st SIR Block** | Number of user sessions that fit into the primary user control block (first `USERS` subparameter in `NCMDIR` macro; see *USERS - Session Information Record**). |
| **Slots in SIR Block Extns** | Number of user sessions that fit into a secondary user control block (second `USERS` subparameter in `NCMDIR` macro; see *USERS - Session Information Record**). |
| **VSAM Roll File Slots** | Number of VSAM roll files to check (ROLLFLS). |
| **Possible Roll Facilities** | Number of VSAM roll files plus two for CICS (`MAINSTOR` and `TEMPSTOR`). |
| **Thread Groups** | Number of thread groups determined by evaluating all `NCMTGD` macro specifications at system startup. See also *NCMTGD Macro Parameters**. |
| **System Recoveries** | Number of corrections of statistics counts and/or control block chain. |
| **Size of DIR Extension (B)** | Number of bytes used at system startup for thread control blocks and VSAM roll file online directories. |
| **Operating System Host ID** | Name of the operating system image. |
| **Assembled Last** | Date and time when the system directory source module was last assembled. |
| **CICS System ID** | ID of the CICS region. |
| **Available Resources:**<br><br>**Swap Pool**<br>**Local Buff. Pool**<br>**Sort Buffer Pool**<br>**DL/I Buffer Pool**<br>**Edit Buffer Pool**<br>**Monitor Pool**<br>**RNM Buffer Pool** | Resources available in the current NCI system environment: swap pools, Natural buffer pools, monitor buffer pools and RNM buffer pools.<br><br>Type, size (in KB) and location (`below` or `above` the 16 MB line) of all buffer pools supported. |
| **Max Thread Size** | Largest thread size across all valid thread groups. |
| **VSAM Roll Files** | Number of VSAM roll files. |
| **Main/Aux TempStor** | Indicates whether CICS main or auxiliary temporary storage is available for the Natural/CICS roll facilities. |
| **Session Logging** | Indicates whether the Natural/CICS log destination is defined in the CICS DCT (destination control table) and whether the log destination is available. The log destination for sessions is defined with the `LOGDEST*` parameter. |
| **Message Logging** | Indicates whether the Natural/CICS error message log destination is defined in the CICS DCT and whether the log destination is available.<br><br>The log destination for messages is defined with the `MSGDEST*` parameter. |

| Field | Explanation |
|---|---|
| **Message Switching** | Indicates whether the message switching transaction ID is defined in CICS and whether the transaction ID is available. The transaction for switching messages is defined with the `MSGTRAN`* parameter.<br><br>If a transaction ID is not available, a SYSTP session flush (see ***Flush session*** in *Natural User Sessions*) is not possible. |
| **Console Terminal** | Indicates whether the CICS console terminal for Natural/CICS is available. |

* This parameter is contained in the `NCMPRM` macro of the `NCIPARM` parameter module, or the `NTCICSP` macro of the Natural parameter module depending on the Natural CICS Interface version installed. The macros are described in the *TP Monitor Interfaces* and *Parameter Reference* documentation, respectively.

## NCI Generation Options

This function is used to display generation parameter settings for Natural running under CICS. The values of these parameters are determined in the macro `NCMPRM` of the `NCIPARM` parameter module, or the `NTCICSP` macro of the Natural parameter module (depending on the Natural CICS Interface version installed) and created during installation.

When you invoke this function, the **Generation Options** screen appears for the current CICS region (as indicated by the CICS ID in the screen title). This screen displays an overview of the generation option settings for Natural.

Behind each parameter setting in the **Generation Options** screen is a parameter of the `NCMPRM` or `NTCICSP` macro. These parameter names can be viewed by pressing PF10. Press PF10 to toggle between the screen containing the parameter names and explanations of the parameters.

**Related Topics:**

- *Installing Natural CICS Interface* in the *Installation for z/OS* and *Installation for z/VSE* documentation
- *NCMPRM Macro Parameters* in the *TP Monitor Interfaces* documentation
- `NTCICSP` macro in the *Parameter Reference* documentation

# Natural Thread Group Definitions

This function is used to display Natural thread group definitions.

When you invoke this function, the **Natural Thread Group Definitions** screen appears for the current CICS region (as indicated by the CICS ID in the screen title). This screen displays the following information:

| Column | Explanation |
|---|---|
| **Grp No.** | Thread group number. |
| **Group Type** | Type of group definition:<br>`SHR` — Permanent storage threads to be used for thread group.<br>`GETM` — Storage threads allocated by using `GETMAIN`.<br>`none` — No threads to be used; all Natural storage requests are passed to CICS.<br>`Alias` — Thread group redefinition to assign other primary roll facility triggered by transaction ID/task request key. |
| **Roll Fac.** | Primary roll facility assigned:<br>`VSAM`, `Aux` (auxiliary temporary storage), `Main` (main temporary storage) or `none` (no roll facility assigned). |
| **Thread Size** | Thread storage `GETMAIN` size (for thread group types `GETM` and `SHR`). |
| **TCBs** | Maximum number of Natural sessions concurrently active in this thread group. |
| **Transaction IDs / Task Request Keys** | As defined in the CICS transaction definitions for Natural. |

## Commands for Natural Thread Group Definitions

In addition to the commands described in *Using SYSTP Utility Screens*, the **Natural Thread Group Definitions** screen provides the following PF keys and corresponding line commands:

| PF Key | Line Command | Function |
|---|---|---|
| PF4 | S | Displays thread group definitions for the thread group marked with the cursor/command. |
| PF10 | G | Displays **Natural storage threads** (see the relevant section) associated with the thread group marked with the cursor/command. |

## Own Natural User Session

This function invokes the **Natural User Session Statistics** screen described in ***Natural User Session Statistics***.

## CICS Task Information

This function invokes the **SYSTP Task Information** window, which displays status information about the Natural task in a CICS environment.

## System Administration Facilities

This function is used to access facilities for debugging and tracing.

When you invoke this function, a menu appears with the following functions:

- Trace Facilities
- Debugging Facilities
- System Snapshot for Logging
- Reset System Highwater Marks
- Common Dynamic Parms Control Information
- Applied NCI Source Changes

- Applied NCI Zaps

### Trace Facilities

This function reserved for internal use by Software AG personnel only.

### Debugging Facilities

This function reserved for internal use by Software AG personnel only.

### System Snapshot for Logging

This function provides complete SYSTP batch reports (see also ***SYSTP in Batch for CICS Sessions***) with information about all SCP facilities, regardless of whether they have been used or not. Such facilities are:

- Thread groups
- `TYPE=SHR` threads
- Roll facilities

All this information is logged to the Natural/CICS log file, if available.

### Reset System Highwater Marks

This function comprises the system snapshot function previously described. In addition, all system highwater marks can be reset, for example:

- The number of user sessions.
- Every thread group and roll facility.
- The number of UCB block extensions.
- The amount of storage.
- All thread groups and `TYPE=SHR` threads.
- All wait queue values and counts.
- All roll facility roll counts.

### Common Dynamic Parms Control Information

This function displays common dynamic profile parameters as retrieved from the `PRMDEST` destination, if available. See also *PRMDEST - Name of the Natural CICS Profile Parameter Input Destination* described in the *TP Monitor Interfaces* documentation.

### Applied NCI Source Changes

This function invokes the **Applied NCI Source Changes** screen for the current CICS region (as indicated by the CICS ID in the screen title). This screen displays the numbers of all source changes that have been applied to the current Natural TP environment.

### Applied NCI Zaps

This function invokes the **Applied NCI Zaps** screen for the current CICS region (as indicated by the CICS ID in the screen title). This screen displays the numbers of all Zaps that have been applied to the current Natural TP environment.

# 123 SYSTP Functions under IMS TM


- Broadcasting ..... 1056
- Display Environment Data ..... 1056
- Monitoring ..... 1057
- Applied NII Zaps ..... 1057

The SYSTP utility provides functions that are specific to IMS TM.

**To invoke specific SYSTP functions under IMS TM**

- In the **Code** field of the SYSTP **Main Menu**, enter `E` for **Environment-Dependent Functions**.

  From the **NII Menu** displayed then, you can select the functions explained in this section.

**Note:** In the remainder of this section, the Natural openUTM Interface is also referred to as NII.

## Broadcasting

This function is used to broadcast messages to specific user groups in an IMS environment.

When you invoke this function, the **Broadcasting Menu** appears from which you can select the following functions:

- **Create Broadcast Messages**
- **List all Broadcast Messages**

For further information about the broadcasting function, see the section *Natural under IMS TM - Special Functions* in the *TP Monitor Interfaces* documentation.

## Display Environment Data

This function is used to display environmental data on the Natural IMS TM Interface.

When you invoke this function, the **Environment Table** screen appears for the environment table used by the current Natural session. The screen displays the current parameter settings of the Natural openUTM Interface.

The parameters cannot be updated. For more information about IMS parameters, see the section *Natural under IMS TM* in the *TP Monitor Interfaces* documentation.

## Monitoring

This function is used to display monitoring data about Natural user sessions that run under the same Natural subsystem.

When you invoke this function, the **Monitoring** screen appears where you can select the following functions to display monitoring data about user sessions:

- **Active Sessions**
  Displays all active Natural sessions that run under IMS TM.
- **Suspended Sessions**
  Displays all Natural sessions that are currently suspended under IMS TM.
- **User Selection**
  Invokes a window where you can specify selection criteria to display particular Natural user sessions only.

## Applied NII Zaps

This function invokes the **Applied NII Zaps** screen, which displays the numbers of all Zaps that have been applied to the current Natural TP environment.

# 124 SYSTP Functions under TIAM and UTM


- P-Key Utility ..... 1060
- Show Common Memory Pools ..... 1064

The SYSTP utility provides functions that are specific to TIAM and UTM.

**To invoke specific SYSTP functions under TIAM and UTM**

- In the **Code** field of the SYSTP **Main Menu**, enter `E` for **Environment-Dependent Functions**.

  From the menu displayed then, you can select the functions explained in this section.

## P-Key Utility

This function supports the loading of programmable P keys on terminal devices of the 975*n* series (types 974*n*, 975*n* and 976*n*).

You can load either the standard Natural key settings (function-key mode `KN`, `KO` or `KS`) to the keys `P1` to `P20`, or user-defined values to individual keys. See also *Function Keys Supported under BS2000* in the *Operations* documentation.

This function invokes the **P-Key Utility** menu, which provides the following menu:

```
15:54:05             ***** NATURAL SYSTP UTILITY *****             1998-03-25
User VR000001                -  P-Key Utility -                TID 0709


                      Code   Function           Parameter

                       KU    Load User Values   A,H
                       KS    Set KS Mode        L,N
                       KN    Set KN Mode        L,N
                       KO    Set KO Mode        L,N
                       KF    Load F1 - F20
                       ?     Help
                       .     Exit


                Code .. __              Parameter  A




Select function.
Command ===>
Enter-PF1---PF2---PF3---PF4---PF5---PF6---PF7---PF8---PF9---PF10--PF11--PF12---
      Help  Menu  Exit  KU    KS    KSN   KN    KNN   KO    KON   KF    Canc
```

You enter a function code and an optional parameter code in this menu. The valid parameter codes for a function are listed to the right of the function. The codes have the following meaning.

| Parameter | Meaning |
|---|---|
| A | Values are entered in alphanumeric format. |
| H | Values can be entered in alphanumeric or hexadecimal format. |
| L | Load option. Mode is set and P keys are loaded. |
| N | No-load option. Mode is set, but P keys are not loaded. |

This section covers the following topics:

- Load User Values
- Load User Values with LPFSUP01 Interface
- Set Key-Assignment Mode
- Load Send-Key Codes to P Keys

## Load User Values

This function allows you to load your own values to the keys P1 to P20 (for example, terminal functions, send codes). Values can be entered either in alphanumeric or hexadecimal format. You choose the required format by entering A (alphanumeric format) or H (hexadecimal or alphanumeric format) in the **Parameter** field of the menu. When you invoke the **Load User Values** function (by pressing PF4 or by entering function code KU in the **P-Key Utility** menu), the following screen is displayed:

```
15:55:41             ***** NATURAL SYSTP UTILITY *****             1998-06-25
User VR000001                - Edit User Keys -                 TID VR000001

P1:             ________________________
                4040404040404040404040404040404040404040404040404040
P2:             ________________________
                4040404040404040404040404040404040404040404040404040
P3:             ________________________
                4040404040404040404040404040404040404040404040404040
P4:             ________________________
                4040404040404040404040404040404040404040404040404040
P5:             ________________________
                4040404040404040404040404040404040404040404040404040
P6:             ________________________
                4040404040404040404040404040404040404040404040404040
P7:             ________________________
                4040404040404040404040404040404040404040404040404040
P8:             ________________________
                4040404040404040404040404040404040404040404040404040
 Function ( + / - / L / . )   _                       Mode:    ALPHA
 Select function or fill up values for P-Keys.
 Command ===>
Enter-PF1---PF2---PF3---PF4---PF5---PF6---PF7---PF8---PF9---PF10--PF11--PF12---
            Menu  Exit  Load               -     +                      Canc
```

The **Mode** field is set to HEX or ALPHA depending on the parameter you specify when invoking the function. You can switch modes by replacing the current value by A (for ALPHA) or H (for HEX).

- In ALPHA mode, you can use the left half of the screen to enter an alphanumeric value next to the key you wish it to be loaded to.
- In HEX mode, you can also assign a value to a key in hexadecimal form on the right half of the screen.

For each P key, enter an alphanumeric value in the empty input field or a hexadecimal value in the line below it (for **Parameter** value A, the hexadecimal field is input blocked).

If no value is specified for a key, the standard Natural key setting (function-key mode KN, KO or KS) applies for this key; thus, it is possible to have a mixed P-key usage; that is, some keys with user-defined functions, others with the standard Natural functions.

Load the values by pressing PF4 or by entering L in the **Function** field.

Page the screen to additional P keys by pressing PF8 or by entering a plus (+) sign in the **Function** field.

**Note:** Natural automatically converts all binary values which are smaller than H'40' to H'6F' (= question mark). So, before any binary values smaller than H'40' can be loaded, the macro NTTAB (translation table) has to be changed accordingly so as to avoid this automatic conversion. This is particularly important for H'27' (= ESCAPE) and H'19' (= Endemarke). For detailed information, see *TAB - Standard Output Character Translation* in the *Parameter Reference* documentation.

### Load User Values with LPFSUP01 Interface

The **Load User Values** function is also available to user applications as an application programming interface (API). The API consists of the Natural subprogram LPFSUP01, which performs the loading of the keys. LPFSUP01 is supplied in the system library SYSEXTP and can be copied into user libraries or steplibs.

#### To call LPFSUP01

- Issue the following statement:

```
CALLNAT 'LPFSUP01' P-VALUE(*)
```

where P-VALUE must be defined as an array: (A24/20).

**Example:**

```
DEFINE DATA LOCAL
     1 P-VALUE (A24/20)
  END-DEFINE
  * LOAD '/STA L EM DUE1' TO P1, '/STA P EM DUE1' TO P4
  COMPRESS '/STA L' h'192786' INTO P-VALUE(1)
  COMPRESS '/STA P' h'192786' INTO P-VALUE(4)
  CALLNAT 'LPFSUP01' P-VALUE(*)
  END
```

See also the example program LPFEXAM1 in the system library SYSEXTP.

### Set Key-Assignment Mode

The following functions are used to set key-assignment modes on terminal devices of the 975*n* series (types 974*n*, 975*n* and 976*n*):

| Mode | Function |
|---|---|
| **Set KS Mode** | Executes the terminal command `%KS`* and is invoked by either pressing PF5 or entering function code `S` in the **P-Key Utility** menu. |
| **Set KN Mode** | Executes the terminal command `%KN`* and is invoked by either pressing PF7 or entering function code `N` in the **P-Key Utility** menu. |
| **Set KO Mode** | Executes the terminal command `%KO`* and is invoked by either pressing PF9 or entering function code `O` in the **P-Key Utility** menu. |

* described in the *Terminal Commands* documentation

For detailed explanations of key-assignment modes, see *Natural under BS2000* in the *Operations* documentation.

### Load Send-Key Codes to P Keys

The **Load F1 - F20** function is used to load specific send-key (F) codes `F1` to `F20` to the keys P1 to P20. The function is similar to the key assignment mode `KN`, except that F codes can be selected individually.

When this function is invoked, the following screen appears:

```
15:56:34            ***** NATURAL SYSTP UTILITY *****             1998-06-25
User VR000001                 - Load F-Codes -                 TID VR000001




 P01   _         P02   _         P03   _         P04   _         P05   _

 P06   _         P07   _         P08   _         P09   _         P10   _

 P11   _         P12   _         P13   _         P14   _         P15   _

 P16   _         P17   _         P18   _         P19   _         P20   _



Mark P-Key to be loaded with F-Code
Command ==>
Enter-PF1---PF2---PF3---PF4---PF5---PF6---PF7---PF8---PF9---PF10--PF11--PF12---
Load        Menu  Exit                                                   Canc
```

To load P keys with F codes, mark the appropriate keys and press ENTER. Only the keys which are marked are loaded with F codes. Other P keys retain their original values.

## Show Common Memory Pools

This function displays a list of all common memory pools used in Natural.

The individual items of information shown for each common memory pools are explained in the online help about this function.

# 125 SYSTP in Batch for CICS Sessions


- Invoking SYSTP in Batch Mode ..... 1066
- Evaluating the Log File ..... 1066

The SYSTP utility can also be used to obtain statistical data on Natural/CICS sessions in batch mode.

The Natural log file into which the statistical data about Natural/CICS sessions is written must be assigned to the Natural batch job as Work File 1 (that is, via `CMWKF01`). It must also be defined to the online system, which means in the CICS DCT (destination control table); see the `LOGDEST` parameter of the `NCMPRM` or `NTCICSP` macro (depending on the Natural CICS Interface version installed) described in the *TP Monitor Interfaces* or *Parameter Reference* documentation, respectively.

## Invoking SYSTP in Batch Mode

**To invoke the SYSTP utility in batch mode**

- In the batch job, specify either of the following commands:

  ```
  SYSTP xxx
  ```

  or

  ```
  LOGON SYSTP
  SYSBATCH xxx
  ```

  where *xxx* indicates what kind of data is to be processed; for example: *xxx*=`nci` indicates that the data is collected by a Natural/CICS online system.

## Evaluating the Log File

Data is written into the Natural log file when Natural is initialized or reset, and when a Natural session is terminated.

The Natural CICS Interface writes the following records into the Natural log file:

- A start log record whenever the Natural environment is initialized or reset.
- A session log record whenever a Natural session is terminated.

When a Natural environment is initialized, a system ID is written into the system control block. This system ID also belongs to all log records. Therefore, a Natural log file can be shared by several Natural/CICS online environments.

The information logged serves to keep track of the usage of the Natural/CICS online environment. Therefore, most of the information refers to facilities of the Natural environment. The log file is not intended to be an accounting or monitoring tool that refers to facilities of CICS.

Based on the system ID, several reports are created with data related to a Natural session:

- Log file data listed in chronological order, which means that session log records are sorted by session end date and time.
- Statistics about how the Natural environment was set up and used.
- Statistics about thread groups (if used).
- Statistics about program storage threads (if used).
- Statistics about roll facilities (if used).

This set of reports is created for all Natural environments with records about Natural/CICS sessions in the Natural log file.

**Note:** The session termination log records, of course, reflect only resources which have been used by the corresponding sessions. Therefore, these records may not reflect the full Natural environment. Reports of a full Natural environment can be obtained by making a snapshot of the whole Natural environment using the **System Administration Facilities** function (see the relevant section).

# Index


## A

activate
    utility, 3
advanced-user mode
    Object Handler, 113
API
    locate and test API of add-on product, 309
    locate and test Natural API, 521
application
    define command-driven navigation system, 681

## B

batch condition code
    Object Handler, 277
buffer
    calculate size with SYSRPC utility, 991

## C

code page
    show information about code page, 439

## D

DDM
    load, 68
direct command
    Object Handler, 217

## E

error message
    load, 69

## F

find
    Object Handler, 131
FNAT
    read messages with USR0020P, 517
fSYSCP utility
    use, 439
FUSER
    read messages with USR0020P, 517

## I

initialize
    Natural Security environment with INPL, 71
INPL file
    scan, 71
INPL utility
    use, 61
invoke
    Profiler utility, 798
    SYSAPI utility, 310
    SYSCP utility, 440
    SYSEXT utility, 524

## L

library
    example library for new features, 537
    load, 68
load
    all objects, 69
    DDM, 68
    error message, 69
    library, 68
    object, 61
    resource, 61

## N

Natural Security
    force environment initialization with INPL, 71
navigation system
    define with SYSNCP utility, 681

## O

object
    find with Object Handler, 131
    load, 61
    load all, 69
    scan, 61
Object Handler
    advanced-user mode, 113
    direct command, 217
    main functions, 103
object list
    Object Handler, 247
option-setting

Object Handler, 259

## P

parameter-setting
Object Handler, 253
ping server
SYSRPC utility, 999
profile setting
Object Handler, 285
Profiler utility
invoke, 798
terminate, 798

## R

resource
load, 61
scan, 61

## S

scan
INPL file, 71
object, 61
resource, 61
select-clause
Object Handler, 223
server
ping with SYSRPC utility, 1002
terminate with SYSRPC utility, 1004
server command execution
SYSRPC utility, 999
service directory
maintain with SYSRPC utility, 957
settings
Object Handler, 187
show
all code pages with SYSCP utility, 454
show information
about code page, 439
size requirement
calculate with SYSRPC utility, 991
stub
generate with SYSRPC utility, 973
SYSAPI utility
invoke, 310
terminate, 310
use, 309
SYSCP utility
invoke, 440
show all code pages, 454
terminate, 440
SYSEXT utility
invoke, 524
terminate, 524
use, 521
SYSEXV
example library for new features, 537
SYSNCP utility
use, 681

## T

terminate
Profiler utility, 798
SYSAPI utility, 310
SYSCP utility, 440
SYSEXT utility, 524
terminate server
SYSRPC utility, 999
test
API of add-on product, 309
Natural API, 521
tools
Object Handler, 281

## U

use
INPL utility, 61
SYSAPI utility, 309
SYSCP utility, 439
SYSEXT utility, 521
SYSNCP utility, 681
user exit routine
Object Handler, 277
USR0020P
read messages from the FNAT or FUSER, 517
utility
activate, 3
overview, xvii

## W

work file
Object Handler, 213
workplan
Object Handler, 133, 203