Natural for UNIX

ツールおよびユーティリティ

バージョン 8.4.1

2017 年 10 月

ADABAS & NATURAL

このマニュアルは Natural バージョン 8.4.1 およびそれ以降のすべてのリリースに適用されます。

このマニュアルに記載される仕様は変更される可能性があります。変更は以降のリリースノートまたは新しいマニュアルに記述されます。

Copyright © 1992-2017 Software AG, Darmstadt, Germany and/or Software AG USA, Inc., Reston, VA, United States of America, and/or their licensors.

The name Software AG, webMethods and all Software AG product names are either trademarks or registered trademarks of Software AG and/or Software AG USA, Inc. and/or their licensors. Other company and product names mentioned herein may be trademarks of their respective owners.

Software AG およびその子会社が所有する登録商標および特許の詳細については、 http://documentation.softwareag.com/legal/ を確認してください。

本ソフトウェアの一部にはサードパーティ製製品が含まれています。サードパーティの著作権表示およびライセンス規約については『License Texts, Copyright Notices and Disclaimers of Third-Party Products』を参照してください。このドキュメントは製品ドキュメントセットの一部であり、http://documentation.softwareag.com/legal/上、またはライセンス製品のルートインストールディレクトリ内にあります。

本ソフトウェアの利用は、Software AG のライセンス規約に則って行われるものとします。ライセンス規約は製品ドキュメントセット内、http://documentation.softwareag.com/legal/上、またはライセンス製品のルートインストールディレクトリ内にあります。

ドキュメント IDは: NATUX-NNATUTILITIES-841-20200614JA

目次

前書き xi
1 1
表記規則 2
オンライン情報 2
データ保護 3
I ユーティリティのアクティブ化 5
2 ユーティリティのアクティブ化 7
II FTOUCH ユーティリティ 9
3 FTOUCH ユーティリティ 11
FTOUCH ユーティリティの使用 12
ftouch の構文 13
例：ftouch 17
III INPL ユーティリティ 19
4 INPL ユーティリティ 21
INPL ユーティリティの概要 22
Load Libraries Only 26
Load DDMs Only 27
Load Error Messages Only 28
Load All Objects 28
Scan INPL File 29
Natural Security Recover 30
ユーザー出口ルーチン 30
IV Natural Termcap（NATTERMCAP）ユーティリティ 33
5 Natural Termcap（NATTERMCAP）ユーティリティ 35
全体的な設定 36
Natural Termcap ユーティリティの呼び出し 38
Natural Termcap ユーティリティの終了 42
Terminal Copy Capabilities 43
キー定義 44
［File］メニュー 44
［Edit］メニュー 45
［Search］メニュー 46
［Test］メニュー 46
［Options］メニュー 47
［Help］メニュー 47
端末機能 - 概要 47
端末機能 - 名前別 54
V オブジェクトハンドラ 59
6 オブジェクトハンドラの全般的な情報 61
オブジェクト転送の原理 62
オブジェクトハンドラの呼び出し 64
バッチまたはダイレクトコマンドの呼び出し 65
Natural プログラムからのオブジェクトハンドラコマンドの発行 66

Natural Security 66
標準の PF キー 67
FDDM システムファイルの使用 67
7 機能 69
8 ウィザード 71
手順 1 - プロシージャの開始 72
手順 2 - ワークファイルへのオブジェクトのアンロード／ワークファイルからのオブジェクトのロード／ワークファイルのオブジェクトのスキャン 73
手順 3 - パラメータの設定 75
手順 4 - オブジェクトの選択 76
手順 5 - 処理の実行 78
手順 6 - 処理の続行 78
9 上級ユーザー 79
上級ユーザーのアクティブ化 80
オブジェクトの処理 80
10 コンパクトモード 83
コンパクトモードの選択方法 84
コンパクトモードでの命令の処理のされ方 84
11 ロードの再スタート 89
12 ビュー 91
Natural ライブラリオブジェクト 92
Natural システムエラーメッセージ 93
Natural コマンドプロセッサソース 94
FDT 95
13 検索 97
14 管理 99
ワークプランライブラリ内の利用可能なワークプランのリストの作成 100
新しいワークプランの作成 102
ワークプランライブラリの変更 104
15 システムファイルの選択 107
16 ライブラリの選択 109
17 システムエラーメッセージの選択 111
列およびコマンド 112
18 オブジェクトの選択 115
リスト画面の列およびコマンド 116
19 オブジェクト指定 119
20 オブジェクト指定 - ワークファイルのすべてのオブジェクト 121
21 オブジェクト指定 - Natural ライブラリオブジェクト 123
Natural ライブラリオブジェクト 124
Natural ライブラリオブジェクトの詳細 125
Natural ライブラリオブジェクトのプロパティ 128
Natural ライブラリオブジェクトの例外 129
Natural ライブラリオブジェクトの例外のプロパティ 130
22 オブジェクト指定 - Natural システムエラーメッセージ 133

Natural システムエラーメッセージ 134
Natural システムエラーメッセージの詳細 134
Natural システムエラーメッセージの例外 135
23 オブジェクト指定 - Natural コマンドプロセッサ 137
Natural コマンドプロセッサ 138
Natural コマンドプロセッサソースの例外 139
24 オブジェクト指定 - Natural 関連オブジェクト 141
Natural 関連オブジェクト 142
Natural 関連オブジェクトの詳細 143
Natural 関連オブジェクトの例外 144
25 オブジェクト指定 - 外部オブジェクト 147
外部オブジェクト 148
外部オブジェクトの詳細 149
外部オブジェクトの例外 150
26 オブジェクト指定 - FDT 151
27 SELECTION または LIST ワークプランの使用 153
28 設定 155
設定画面のフィールド 156
その他のオプションの設定 159
グローバルパラメータの設定 167
29 「ワークプラン」 171
ワークプランの作成、選択、および変更 172
ワークプランの内容 172
ワークプランの例 173
ワークプランの参照 174
30 名前と日時の指定 177
Name 178
日付 179
時刻 180
31 ワークファイル 181
ワークファイルの割り当て 182
ワークファイルフォーマット 183
32 ダイレクトコマンド 187
33 基本的なコマンド構文 189
34 select-clause 193
select-clause の構文 194
SELECTION または LIST ワークプラン 194
Natural ライブラリオブジェクトと DDM の選択 195
Natural 関連オブジェクトの選択 201
Natural システムエラーメッセージの選択 203
Natural コマンドプロセッサソースの選択 204
外部ファイルの選択 206
FDT の選択 207
アプリケーションの選択 208
削除命令のためのオブジェクト選択 211

35 オブジェクトリスト - LIST ワークプラン 215
object-type-and-location の構文 216
object-name-description の構文 217
オブジェクトリストの例 218
36 parameter-setting 221
parameter-clause の構文 222
parameter-clause のキーワードの説明 223
37 option-setting 227
option-setting の構文 228
option-setting のキーワードの説明 230
38 ダイレクトコマンドの使用例 239
同一プラットフォームのオブジェクトのアンロード 240
異なるプラットフォームのオブジェクトのアンロード 241
内部フォーマットでのオブジェクトのロード 242
転送フォーマットでのオブジェクトのロード 242
39 ナビゲーションと特別な機能のためのコマンド 243
40 バッチのコンディションコードとユーザー出口ルーチン 249
バッチで返されるコンディションコード 250
ユーザー出口ルーチンの適用 250
使用可能なユーザー出口ルーチン 251
41 Tools 253
ステータス 254
最新の結果 254
「トレース」 254
レポート 255
42 プロファイル設定 257
PF キー 259
行コマンド 259
プロファイルパラメータ 260
43 NATUNLD/NATLOAD および SYSTRANS からオブジェクトハンドラへの移行 265
個々のコマンドの変換 266
OBJHAPI を使用した SYSTRANS コマンドの処理 268
サポートされていない SYSTRANS オプション 268
VI SYSERR ユーティリティ 269
44 メッセージの全般的な情報 271
メッセージタイプ 272
メッセージの言語 273
メッセージの発行 274
Natural システムショートメッセージの取得 275
ユーザー定義ショートメッセージの取得 275
メッセージ情報の取得 276
45 SYSERR 呼び出し 277
46 機能 279
メッセージの追加 280

メッセージの削除 283
メッセージの表示 284
メッセージの変更 285
メッセージの出力 287
メッセージのスキャン 289
リストからのメッセージの選択 291
メッセージの他の言語への翻訳 293
47 パラメータ 297
Message Type 298
ライブラリ 298
メッセージ番号 298
Language Codes 298
48 ダイレクトコマンド 299
49 大文字変換 - ERRUPPER 303
50 文字置換 - ERRCHAR 305
51 メッセージファイルとテキストファイルの生成 307
メッセージファイルの保存 308
テキストファイルの作成 308
メッセージファイルの生成 309
テキストファイルの再作成 310
52 さまざまなライブラリのメッセージの管理 311
53 アプリケーションプログラミングインターフェイス USR0020P 313
VII SYSEXT ユーティリティ - Natural アプリケーションプログラミングインターフェイス 315
54 SYSEXT ユーティリティ - Natural アプリケーションプログラミングインターフェイス 317
SYSEXT の概要 318
SYSEXT の呼び出しと終了 319
SYSEXT ユーティリティの使用 321
インターフェイスバージョン 326
予約キーワード 326
Natural API の使用 327
VIII SYSEXV ユーティリティ 329
55 SYSEXV ユーティリティ 331
現行バージョンのプログラム例の実行 332
任意のバージョンのプログラム例の実行 332
PF キー 333
SYSEXV ユーティリティの終了 333
IX SYSMAIN ユーティリティ - オブジェクトメンテナンス 335
56 全般的な情報 337
57 SYSMAIN の呼び出しと終了 339
SYSMAIN の呼び出し 340
SYSMAIN の終了 341
58 機能の説明 343
59 オブジェクトのリストと検索 345

60 オブジェクトのコピー、移動、および名前変更 349
61 オブジェクトの削除 355
62 オブジェクトのインポート 359
63 オブジェクト指定ウィンドウのフィールドの使用 363
名前の範囲の指定 366
置換オプションの使用 367
64 サブプログラムでの SYSMAIN の使用 369
MAINUSER の呼び出しと実行 370
コマンドの使用 370
LIST および FIND コマンド構文 371
COPY および MOVE コマンド構文 372
DELETE コマンド構文 372
RENAME コマンド構文 373
IMPORT コマンド構文 373
where-clause 374
with-clause 374
コマンドのキーワードと変数 374
65 XRef について 381
XREF を N に設定 382
XREF を Y に設定 382
FDIC ファイルセキュリティ 383
XRef 処理のエラー 383
X SYSNCP ユーティリティ 385
66 SYSNCP ユーティリティ 387
UNIX の前提条件 388
SYSNCP ユーティリティについて 388
SYSNCP の呼び出し 394
プロセッサの選択 396
ヘッダーレコード 397
キーワードのメンテナンス 406
機能のメンテナンス 412
ランタイムアクション 417
プロセッサのカタログ化 422
管理者サービス 423
セッションプロファイル 430
XI SYSPCI ユーティリティ - 製品の設定および初期化 435
67 SYSPCI ユーティリティ - 製品の設定および初期化 437
画面を使用したインストール済み製品の設定 438
ダイレクトコマンドデータを使用した SYSPCI ユーティリティの呼び出し 444
XII 447
68 Natural Profiler 449
69 Profiling Natural Applications 451
Introducing Profiling 452
Platform-Specific Profiling 452

Profiling Tools ... 453
Natural Profiler Evaluations ... 455
70 Code Coverage of Natural Applications ... 463
Introducing Code Coverage ... 464
Platform-Specific Code Coverage ... 465
Code Coverage Tools ... 465
Natural Code Coverage Evaluations ... 468
71 Basic Concepts of the Profiler Utility ... 475
Data Consolidation, Code Coverage and Data Processing ... 476
Sampling ... 479
Profiling Long-Running Applications ... 481
Related Topics ... 484
72 Using the Profiler Utility ... 485
Quick Start for Profiling ... 486
Quick Start for Code Coverage ... 489
Prerequisites ... 491
Invoking and Terminating the Profiler Utility ... 492
Syntax and Keywords ... 492
Events and Data Collected ... 496
Starting and Pausing Data Collection ... 500
Consolidating Event Data ... 504
Evaluating Event Data ... 506
Exporting Event Data for MashZone ... 521
Maintaining Profiler Resource Files ... 523
Including Profiler Input from Natural Text Objects ... 525
Event Trace ... 526
Tracing Natural Code Coverage ... 528
Internal Trace ... 529
Profiler Statistics ... 530
73 Natural Profiler MashApp ... 539
Preparing to Use the MashApps ... 540
Preparing the Profiler Data ... 544
Opening the MashApps ... 545
Evaluation Page ... 546
Compare Page ... 555
Properties Page ... 557
Use Cases ... 559
XIII SYSRPC Utility ... 577
74 Invoking and Terminating SYSRPC ... 579
Invoking SYSRPC ... 580
Terminating SYSRPC ... 580
Invoking Online Help ... 581
75 Service Directory Maintenance ... 583
Service Directory Concept ... 584
Invoking Service Directory Maintenance ... 585

Fields on the Service Directory Screen 587
Commands for Service Directory Maintenance 590
76 Replacing Items in the Service Directory 595
Syntax of SYSRPC SM REPLACE 596
77 Generating Interface Objects - General Considerations 599
78 Generating Single Interface Objects with Parameter Specification 601
Using the Interface Object Generation Function 602
Specifying Parameters 604
Examples of Interface Object Generation 607
79 Generating Multiple Interface Objects 611
Using the SYSRPC SGMASS Direct Command 612
Name Specification and Compression 614
80 Calculating Size Requirements 617
Using the SYSRPC CSMASS Direct Command 618
Name Specification and Compression 620
81 Parameter Maintenance 621
Invoking Parameter Maintenance 622
Specifying Profile Parameters 622
82 Server Command Execution 625
Using Server Command Execution 626
Pinging an RPC Server 628
Terminating an RPC Server 630
83 Listing Servers Registered on EntireX Broker 633
Example of an SYSRPC SRVLIST Direct Command 635
Viewing a Server List 635
Viewing Additional Server Information 636
Customizing Server Lists 637
84 Remote Directory Maintenance 639
Using Remote Directory Maintenance 640
Fields on the Editor Screen of a Remote Directory 641
Commands for Remote Directory Maintenance 642
85 Overview of SYSRPC Direct and Batch Commands 645

前書き

このドキュメントでは、Naturalが提供するツールおよびユーティリティの目的と使用方法について説明します。

ユーティリティのアクティブ化	Natural でツールまたはユーティリティが呼び出される方法について説明します。
FTOUCH	ダウンロード済みオブジェクトを Natural で実行できるオブジェクトにします。
INPL	Software AG により提供された Natural オブジェクトおよび共有リソースをロードまたはスキャンします。
Natural プロファイラ	Natural アプリケーションの内部プロセスフローをモニタリングし、アプリケーションのパフォーマンスとコードカバレッジを分析します。
Natural Termcap（NATTERMCAP）ユーティリティ	Natural の端末依存部分に合わせて端末を調整します。
オブジェクトハンドラ	Natural 環境での分散のために、Natural と Natural 以外のオブジェクトを処理します。これは、ソース環境のオブジェクトをワークファイルにアンロードし、それらをワークファイルからターゲット環境にロードすることによって実行されます。
SYSERR	アプリケーション固有のメッセージを作成します。さらに、既存の Natural システムメッセージのテキストを修正するために使用できます（推奨できません）。
SYSEXT	Natural アプリケーションプログラミングインターフェイス（API）を検索します。
SYSEXV	現在の Natural バージョンで採用された新機能の例を提供します。
SYSMAIN	コピー、移動、削除、インポートなどのオブジェクト操作を Natural で実行します。
SYSNCP	Natural アプリケーションに対するコマンド方式のナビゲーションシステムを定義します。
SYSPCI	初回インストール後に製品を構成して初期化します。
SYSRPC	Natural RPC（リモートプロシージャコール）環境を確立し、メンテナンスします。

1

- 表記規則 2
- オンライン情報 2
- データ保護 3

表記規則

規則	説明
太字	画面上の要素を表します。
モノスペースフォント	`folder.subfolder:service` という規則を使用して webMethods Integration Server 上のサービスの保存場所を表します。
大文字	キーボードのキーを表します。同時に押す必要があるキーは、プラス記号 (+) で結んで表記されます。
斜体	独自の状況または環境に固有の値を指定する必要がある変数を表します。本文で最初に出現する新しい用語を表します。
モノスペースフォント	入力する必要があるテキストまたはシステムから表示されるメッセージを表します。Program code.
{ }	選択肢のセットを表します。ここから1つ選択する必要があります。中カッコの内側にある情報のみを入力します。{ } 記号は入力しません。
|	構文行で相互排他的な2つの選択肢を区切ります。いずれかの選択肢を入力します。| 記号は入力しません。
[ ]	1 つ以上のオプションを表します。大カッコの内側にある情報のみを入力します。[ ] 記号は入力しません。
...	同じ種類の情報を複数回入力できることを示します。情報だけを入力してください。実際のコードに繰り返し記号 (...) を入力しないでください。

オンライン情報

Software AG マニュアルの **Web** サイト

マニュアルは、Software AG マニュアルの Web サイト (**http://documentation.softwareag.com**) で入手できます。このサイトでは Empower クレデンシャルが必要です。Empower クレデンシャルがない場合は、TECHcommunity Web サイトを使用する必要があります。

Software AG Empower 製品のサポート **Web** サイト

もしまだEmpowerのアカウントをお持ちでないのなら、こちらへempower@softwareag.com電子メールにて あなたのお名前、会社名、会社の電子メールアドレスをお書きの上、アカウントを請求してください。

いったんアカウントをお持ちになれば、Empower **https://empower.softwareag.com/** のeServiceセクションにてサポートインシデントをオンラインで開くことができます。

製品情報は、Software AG Empower 製品のサポート Web サイト (**https://empower.softwareag.com**) で入手できます。

機能および拡張機能に関するリクエストの送信、製品の可用性に関する情報の取得、製品のダウンロードを実行するには、Products に移動します。

修正に関する情報を取得し、早期警告、技術論文、Knowledge Base の記事を読むには、**Knowledge Center** に移動します。

もしご質問があれは、こちらの**https://empower.softwareag.com/public_directory.asp** グローバルサポート連絡一覧の、あなたの国の電話番号を選んで、わたくし共へご連絡ください。

Software AG TECHcommunity

マニュアルおよびその他の技術情報は、Software AG TECHcommunity Web サイト (**http://techcommunity.softwareag.com**) で入手できます。以下の操作を実行できます。

- TECHcommunity クレデンシャルを持っている場合は、製品マニュアルにアクセスできます。TECHcommunity クレデンシャルがない場合は、登録し、関心事の領域として [マニュアル] を指定する必要があります。
- 記事、コードサンプル、デモ、チュートリアルにアクセスする。
- Software AG の専門家によって承認されたオンライン掲示板フォーラムを使用して、質問したり、ベストプラクティスを話し合ったり、他の顧客が Software AG のテクノロジをどのように使用しているかを学んだりすることが可能です。
- オープンスタンダードや Web テクノロジを取り扱う外部 Web サイトにリンクできます。

データ保護

Software AG製品は、EU一般データ保護規則(GDPR)を尊重した個人データの処理機能を提供します。 該当する場合、適切な手順がそれぞれの管理ドキュメントに記載されています。

I ユーティリティのアクティブ化

2 ユーティリティのアクティブ化

Natural では、FNAT システムファイル内の対応するユーティリティライブラリにログオンせずに Natural ユーティリティを呼び出します。この結果、Natural では、グローバルデータエリア（GDA）またはアプリケーションに依存しない変数（AIV）、あるいはその両方が維持されます。現在のユーザーライブラリおよび設定は維持されます（GDA または AIV、あるいはその両方をリセットするには、『*パラメータリファレンス*』のプロファイルパラメータ FREEGDA を参照）。

アプリケーション環境の設定を維持するには、ユーティリティライブラリにログオンしないでください。代わりに、ユーティリティに対応する Natural システムコマンドを使用してユーティリティを呼び出します。

ユーティリティが終了すると、ユーティリティを呼び出したライブラリに戻ります。ただし、ユーティリティを呼び出す前にユーティリティライブラリに明示的にログオンした場合は、ユーティリティの終了後、その（ユーティリティ）ライブラリから移動しません。

例外：

オブジェクトソースはアクティブなライブラリ内でのみ編集できるため、ユーティリティ SYSEXT および SYSEXV では、対応するユーティリティライブラリへの暗黙的なログオンを実行します。

Natural Security で Natural ユーティリティの使用を制御する方法の詳細については、『*Natural Security*』ドキュメントの「*Protecting Utilities*」セクションを参照してください。

II FTOUCH ユーティリティ

3 FTOUCH ユーティリティ

- FTOUCH ユーティリティの使用 12
- ftouch の構文 13
- 例：ftouch 17

FTOUCHユーティリティは、ダウンロード済みのオブジェクトをNaturalで実行可能なオブジェクトにする場合に使用します。このためには、オブジェクトをNaturalシステムファイルFNATまたはFUSERにインポートし、*FILEDIR.SAG*ファイルを更新します。

関連トピック：

- 「*ファイル FILEDIR.SAG*」 - 『*オペレーション*』ドキュメント
- 「*NFS を使用した Natural ライブラリの保存*」 - 『*オペレーション*』ドキュメント
- 「*Natural 生成プログラムの転送*」 - 『*プログラミングガイド*』

FTOUCH ユーティリティの使用

このセクションでは、FTOUCH ユーティリティの実行手順について説明します。

注意: 角カッコ（[ ]）で囲まれている部分はオプションです。太字は実際の値であり、表示されているとおりに入力する必要があります。

手順 3.1. FTOUCH ユーティリティを実行するには

1 オペレーティングシステムのコマンドプロンプトを表示します。

2 転送するファイルが（グローバルコンフィグレーションファイルに指定した）FNATまたはFUSER ディレクトリ内に、正しい拡張子付きで存在することを確認します。

3 次の構文を使用して `ftouch` コマンドを入力します。

```
ftouch [fnat=dbid,fnr] [fuser=dbid,fnr][bp=bp-name]
[parm=parm-file] [lib=library-name] [encoding=encoding-name]
[userep=rep-use] [-ignoreext][-v] [-q] [mode] [kind] files
```

または:

移行する場合は、次の構文を使用します。

```
ftouch [fnat=dbid,fnr] [fuser=dbid,fnr]
[parm=parm-file] [lib=library-name][encoding=encoding-name][-q] convert
```

または:

FILEDIR.SAG ファイルをエンディアン変換する場合は、次の構文を使用します。

```
ftouch [fnat=dbid,fnr] [fuser=dbid,fnr]
[parm=parm-file] [lib=library-name][endian=endian-mode]
```

または:

FILEDIR.SAG ファイルにある 1 つまたは複数のオブジェクトをエンコードする場合は、次の構文を使用します。

```
ftouch [fnat=dbid,fnr] [fuser=dbid,fnr]
[parm=parm-file] [lib=library-name][objname=object-name][encoding=encoding-name]
```

または:

FILEDIR.SAG でライブラリの行番号省略状態を設定する場合は、次の構文を使用します。

```
ftouch [fnat=dbid,fnr] [fuser=dbid,fnr]
[parm=parm-file] [lib=library-name][suprln=library-state]
```

ftouch の構文

`ftouch` コマンドには次のオプションがあります。

オプション	説明
`fnat=dbid,fnr`	使用する FNAT システムファイルのデータベース ID とファイル番号を指定します。デフォルト値は、NATPARM パラメータファイルで指定されています。 「*例 2*」も参照してください。
`fuser=dbid,fnr`	使用する FUSER システムファイルのデータベース ID とファイル番号を指定します。デフォルト値は、NATPARM パラメータファイルで指定されています。 「*例 2*」も参照してください。
`bp=bp-name`	使用するバッファプールを指定します。デフォルトの Natural バッファプール NATBP を使用する場合は、*bp-name* を省略できます。使用しない場合は、該当する *bp-name* を指定する必要があります。 **注意:**

オプション	説明
	1. デフォルトの Natural バッファプールがアクティブでないか、指定したバッファプールが存在しない場合、該当するエラーメッセージが表示されます。 2. デフォルトのバッファプール NATBP は削除しないでください。削除すると、Natural が正常に機能しなくなる可能性があります。
parm=*parm-name*	デフォルトの NATPARM パラメータファイルを使用しない場合は、使用するパラメータファイルの名前を指定します。
lib=*library-name*	使用するライブラリを指定します。すでに適切なサブディレクトリで行っている場合は、*library-name* を省略できます。それ以外の場合は、該当する *library-name* を指定する必要があります。
userep=*rep-use*	リポジトリを使用するかどうかを指定します。*rep-use*は、次のいずれかである必要があります。 ON：リポジトリを使用します。 OFF：リポジトリを使用しません。
-v	処理中のディスク I/O に関する統計を表示します。
-q	quiet モードを使用することを示します。エラーメッセージだけが表示され、ステータスメッセージは表示されません。
-ignoreext	ライブラリに含まれるファイルのうち、不明な拡張子を持つファイルを無視するように指定します。-ignoreext オプションは、次のいずれかのオプションと組み合わせて指定できます。 -a -d
mode	プログラミングモードを指定します。sm は、プログラムがストラクチャードモードにあることを指定します。デフォルトは、レポーティングモードです。 「*例 1*」も参照してください。
kind	入力用にサブディレクトリ SRC や GP を指定します。次のいずれかを指定できます。 -s はソースオブジェクト（デフォルト）、 -g カタログ化オブジェクト／生成プログラム -b はソースオブジェクトとカタログ化オブジェクト／生成プログラムの両方。 「*例 2*」も参照してください。
files	処理するファイルを指定します。ファイルごとに *filename.ext* または次を指定できます。

オプション	説明
	`-a` は新しいファイルを追加します。*FILEDIR.SAG* で見つかったディレクトリ内の全ファイルが追加されますが、既存のファイルはそのままです。 `-d` は新しい *FILEDIR.SAG* ディレクトリを作成します。 **注意:** 既存の *FILEDIR.SAG* は削除されて初めから再作成されるため、このオプションを使用する場合は注意が必要です。 「*例 4*」も参照してください。
`-f`	*FILEDIR.SAG* に指定されているオブジェクトのタイムスタンプを更新します。このオプションは、個々のファイルが `files` オプション（上記参照）で指定されている場合にのみ使用できます。
`convert`	古い *FILEDIR.SAG* ファイルが移行されることを示します。バージョン 6.2 より前の Natural バージョンの *FILEDIR.SAG* ファイルが、新しい移植可能な *FILEDIR.SAG* ファイルに変換されます。元の古い *FILEDIR.SAG* ファイルのコピーが、指定されたライブラリのディレクトリ内に *FILEDIR.BCK* ファイルとして保存されます。指定されたライブラリに *FILEDIR.BCK* ファイルがすでに存在する場合、古い *FILEDIR.SAG* は変換されません。 詳細については、『オペレーション』ドキュメントの「*Natural の移植可能なシステムファイル*」を参照してください。 「*例 3*」と「*例 5*」も参照してください。
`sync`	指定したライブラリとシステムファイルが Natural とリポジトリの間で同期を取ることを示します（Windows のみ）。この機能は、*FILEDIR.SAG* が FTOUCH によって変更されるたびに実行される必要があります。 **注意:** `sync` を指定する場合は、`userep=ON` を設定するか、Natural プロファイルパラメータ `USEREP` を `ON` に設定するかのいずれかを行う必要があります。

`encoding=`*`encoding-name`*	*FILEDIR.SAG* に含まれているファイルに使用するコードページを指定します。
	`encoding` オプションは、`ftouch` コマンドに影響を受けるオブジェクトごとに、*FILEDIR.SAG* 内に内部コードページ情報を生成するか、保持されている内部コードページ情報を変更します。このオプションは、ソースオブジェクトまたはカタログ化オブジェクト／生成プログラムの内容を変換しません。
	`encoding` オプションは、次のオプションとともに使用できます。
	`-a`
	`-d`
	`convert`
	`objname`

オプション	説明
	encoding-name は、*NATPARM* パラメータファイルで指定された CP セッションパラメータで使用できるコードページ名です。『パラメータリファレンス』の「*CP - デフォルトのコードページ名*」も参照してください。 「*例 4*」、「*例 5*」、「*例 7*」、「*例 8*」も参照してください。
endian=*endian-mode*	*FILEDIR.SAG* ディレクトリで使用するエンディアンフォーマットを指定します。 endian オプションは、*FILEDIR.SAG* ディレクトリ全体に適用されます。 *FILEDIR.SAG* にファイルを追加する場合または *FILEDIR.SAG* を新規作成する場合、このオプションは適用されません。 *endian-mode* は、次のいずれかです。 BIG　ビッグエンディアンに変換します。 LITTLE　リトルエンディアンに変換します。 DEFAULT　現在のプラットフォームで使用するエンディアンフォーマットに変換します。 「*例 6*」も参照してください。
objname=*object-name*	*FILEDIR.SAG* 内に内部フォーマット情報を保持するオブジェクトを選択します。 objname オプションが適用されるのは、encoding オプションを指定した場合だけです。 *object-name* は、指定された値と同じ名前のオブジェクトをすべて選択します。名前の範囲にはアスタリスク（*）表記を使用できます。 「*例 7*」と「*例 8*」も参照してください。
suprln=*library-state*	指定されたライブラリに行番号省略状態を設定するかどうかを指定します。*library-state* は、次のいずれかである必要があります。 ON　このライブラリに含まれているオブジェクトのソースを保存する場合、ソース行番号は *FILEDIR.SAG* に含まれているファイルに書き込まれません。 OFF　ソース行番号は、*FILEDIR.SAG* に含まれているファイルに書き込まれます。

例：ftouch

次のセクションでは、ftouch コマンドの例を示します。

例 **1**

次のディレクトリに変更します。	*fuser-directory*/TESTLIB/SRC
次のコマンドを入力します。	ftouch sm TESTFILE.NSP

この結果、TESTLIB ライブラリの TESTFILE プログラムが、Natural のストラクチャードモードで使用可能になります。

例 **2**

次のディレクトリに変更します。	*fuser-directory*/MYLIB
次のコマンドを入力します。	ftouch fnat=21,21 fuser=22,22 -b

この結果、MYLIB/SRC ディレクトリと MYLIB/GP ディレクトリにあるすべてのファイルが、Natural のレポーティングモード（デフォルト）で使用可能になります。

例 **3**：

次のディレクトリに変更します。	*fuser-directory*
次のコマンドを入力します。	ftouch lib=MYLIB convert

この結果、新しい移植可能な *FILEDIR.SAG* ファイルが MYLIB ライブラリ用に保存され、古い *FILEDIR.SAG* はこのライブラリ内に *FILEDIR.BCK* ファイルとして保存されます。

例 **4**

次のディレクトリに変更します。	*fuser-directory*
次のコマンドを入力します。	ftouch lib=MYLIB encoding=UTF-8 -a -s

この結果、MYLIB/SRC サブディレクトリから *FILEDIR.SAG* ディレクトリに追加されるすべてのオブジェクトの内部フォーマット情報が、UTF-8 になります。

例 **5**

次のディレクトリに変更します。 *fuser-directory*
次のコマンドを入力します。 `ftouch lib=OLDLIB encoding=windows-1251 convert`

この結果、新しい移植可能な *FILEDIR.SAG* ファイルが OLDLIB ライブラリ用に保存され、*FILEDIR.SAG* ファイルに含まれているすべてのオブジェクトの内部フォーマット情報が windows-1251 に変更されます。

例 **6**

次のディレクトリに変更します。 *fuser-directory*
次のコマンドを入力します。 `ftouch lib=MYLIB endian=BIG`

この結果、MYLIB ライブラリの *FILEDIR.SAG* ファイルはビッグエンディアンに変換されます。MYLIB ライブラリに含まれているすべてのオブジェクトの内部フォーマット情報が BIG に変更されます。

例 **7**

次のディレクトリに変更します。 *fuser-directory*
次のコマンドを入力します。 `ftouch lib=MYLIB objname=MYPROG1 encoding=UTF-8`

この結果、*FILEDIR.SAG* ファイルの MYLIB ライブラリに MYPROG1 が含まれていると、MYPROG1 オブジェクトの内部フォーマット情報は UTF-8 に変更されます。

例 **8**

次のディレクトリに変更します。 *fuser-directory*
次のコマンドを入力します。 `ftouch lib=MYLIB objname=MY* encoding=UTF-8`

この結果、名前が MY で始まり *FILEDIR.SAG* ファイルの MYLIB ライブラリに含まれているすべてのオブジェクトの内部情報が UTF-8 に変更されます。

例 **9**

次のディレクトリに変更します。 *fuser-directory*
次のコマンドを入力します。 `ftouch lib=MYLIB suprln=ON`

この結果、*FILEDIR.SAG* ファイルの MYLIB ライブラリの行番号省略状態が ON に設定されます。

III INPL ユーティリティ

4 INPL ユーティリティ

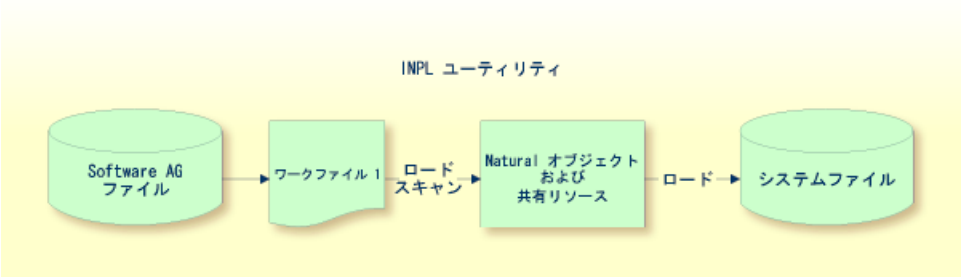
- INPL ユーティリティの概要 22
- Load Libraries Only 26
- Load DDMs Only 27
- Load Error Messages Only 28
- Load All Objects 28
- Scan INPL File 29
- Natural Security Recover 30
- ユーザー出口ルーチン 30

INPL ユーティリティ（Initial Natural Program Load）は、Software AG が提供するファイルから Natural オブジェクトおよび共有リソースをロードまたはスキャンするために使用します。

INPL ユーティリティの概要

INPL ユーティリティは、Software AG が提供する Natural オブジェクトおよび共有リソースを処理します。

次の図は、INPL 機能を簡単な図で示したものです。



Natural オブジェクトおよび共有リソースは、ワークファイル 1 に割り当てられたインストールまたは更新ファイルとして提供されます。INPL ユーティリティは、ワークファイル 1 の Natural オブジェクトおよび共有リソースを Natural システムファイルにロードします。

Natural オブジェクトおよび共有リソースには、Natural システムファイル FNAT および FUSER 内のライブラリに含まれている、カタログ化オブジェクトおよびソースオブジェクトがあります。

INPL ユーティリティは、Natural オブジェクトおよび共有リソースのロード機能だけでなく、ワークファイル 1 に割り当てられたファイルの内容をチェックするスキャン機能、Natural Security 環境を初期化する［**Natural Security Recover**］機能も提供します。

カタログ化オブジェクトを Natural システムファイルにロードする場合、INPL ユーティリティが使用する同一バッファプール内に名前が同じカタログ化オブジェクトが含まれていると、そのオブジェクトのすべてのバッファプールエントリが削除されます。

INPL の実行中にエラーが発生すると、INPL は中断され、Condition Code 40 で異常終了します。

このセクションでは、次のトピックについて説明します。

- 制限
- 特別な場合
- INPL の呼び出し
- 使用可能なオプション

- INPL レポート

制限

処理できるのは、「SAG システム INPL ファイル」 としてマークされたファイルだけです。

特別な場合

INPL を Natural Security 環境で実行する場合、Natural ダイナミックプロファイルパラメータ STACK を使用して INPL コマンドを指定できます。

INPL の呼び出し

手順 4.1. INPL ユーティリティを呼び出すには

1 次の Natural システムコマンドを入力します。

```
INPL
```

次のような INPL メニューが表示されます。

```
11:04:48                ***** NATURAL INPL UTILITY *****            2001-11-09
User: SAG                                                   Library: SYSTEM
                      Code   Function

                        L    Load Libraries Only                              ↩

                        D    Load DDMs Only
                        E    Load Error Messages Only
                        B    Load All Objects
                        S    Scan INPL File
                        R    Natural Security Recover
                        ?    Help
                        .    Exit


         Code ........ B
         Replace ..... Y  (Y/N/O)     Load Except . N  (Y/N)
         DDM Name ....
         Library .....
         Object Name .                Date ........            (YYYY-MM-DD)
         Check Date .. N  (Y/N)       Number ...... 0
         File Type ... D  (D/P)
         Load File ... $NATWORK/SAGLOAD.sag
         Report File . $HOME/report.txt
```

2 対応するファンクションコードを INPL メニューの［**Code**］フィールドに入力することにより、次のいずれかの機能を選択できます。

- ライブラリのみロード
- **DDM** のみロード
- エラーメッセージのみロード
- すべてのオブジェクトをロード
- **INPL** ファイルのスキャン
- **Natural Security** リカバー

これらの機能の詳細については、対応するセクションを参照してください。「*使用可能なオプション*」の説明に従って、残りの入力フィールドを変更または完了してください。

3 ［ENTER］を押してエントリを確定します。

使用可能なオプション

次のセクションでは、INPL で使用するファイルおよび INPL 機能のオブジェクト選択条件として 1 つまたは複数のパラメータを指定できる INPL メニュー上の入力フィールドについて説明します。この選択条件は、［**Code**］フィールドで指定します。使用するパラメータは、関連するドキュメントのセクションで説明するように、機能によって異なります。

フィールド	説明
Replace	処理される Natural オブジェクトおよび共有リソースを、すでにシステムファイルに存在するものと置き換えるかどうかを指定します。 可能な設定： Y すべての既存Naturalオブジェクトおよび共有リソースが置き換えられます。これはデフォルト設定です。 N 既存の Natural オブジェクトおよび共有リソースは*置き換えられません*。 O 指定したオブジェクトのすべてのオーナー情報をリセットします。［**Natural Security Recover**］機能にのみ適用されます。 処理する Natural オブジェクトおよび共有リソースより古い Natural オブジェクトおよび共有リソースだけを置き換える場合は、「**Check Date**」も参照してください。 ［**Natural Security Recover**］機能を使用する場合、このフィールドに「O」オプションを入力すると、指定したオブジェクトのオーナー情報をリセットできます。
DDM Name	DDM の名前または名前の範囲。 アスタリスク（*）で終わる値を入力すると、指定した値で始まる名前を持つDDMが処理されます。アスタリスク（*）のみを入力した場合、またはこのフィールドが空の場合は、すべての DDM が処理されます。

フィールド	説明
Library	ライブラリの名前または名前の範囲。 アスタリスク（*）で終わる値を入力すると、指定した値で始まる名前を持つ各ライブラリが処理されます。［**Object Name**］を指定する場合、ライブラリ名は必須です。
Object Name	Natural オブジェクト（DDM 以外）の名前または名前の範囲。 値がアスタリスク（*）で終わる場合は、指定した値で始まる名前を持つ各オブジェクトが処理されます。 このフィールドが空の場合は、［**Library**］フィールドに指定したライブラリに含まれるすべてのオブジェクトが処理されます。
Check Date	既存の Natural オブジェクトおよび共有リソースをタイムスタンプに基づいて置き換えるかどうかを指定します。 ［**Replace**］を N に指定した場合、このパラメータは無効です。 可能な設定： Y 同じ名前の Natural オブジェクトまたは共有リソースより古いオブジェクトのみが置き換えられます。ロードされる前に保存またはカタログ化されたオブジェクトが古いオブジェクトです。 N すべてのオブジェクトが置き換えられます。これはデフォルト設定です。
Load Except	Natural オブジェクトおよび共有リソースを処理から除外するかどうかを指定します。 このパラメータはエラーメッセージには適用されません。 可能な設定： Y ［**DDM Name**］、［**Library**］、［**Object Name**］の各フィールドで指定したオブジェクトを除くすべての Natural オブジェクトおよび共有リソースが処理されます。 N 除外なしの場合は、すべての Natural オブジェクトおよび共有リソースが処理されます。これはデフォルト設定です。 ロード除外の例： ライブラリ ABC を除くすべてのライブラリがロードされます。 **Code** = L **Library** = ABC 接頭辞が XY 以外のすべての DDM がロードされます。 **Code** = D **DDM Name** = XY* 接頭辞が AB 以外のライブラリに含まれるすべてのオブジェクトと、接頭辞が CD 以外のすべての DDM がロードされます。 **Code** = B **Library** = AB* **DDM Name** = CD*

フィールド	説明
Date	このフィールドに入力した日付以降に保存されたまたはカタログされたNaturalオブジェクトおよび共有リソースだけを処理します。 日付は、*YYYY-MM-DD*（*YYYY* = 年、*MM* = 月、*DD* = 日）の形式で入力する必要があります。
Number	Naturalオブジェクトおよび共有リソースの処理を指定の数に制限します。INPLメニューで指定した選択条件に基づいてロードまたはスキャンされるすべてのオブジェクトがカウントされます。 処理された Natural オブジェクトの数が［**Number**］フィールドに入力した値に達すると、処理は終了し、対応するメッセージが表示されます。
File Type （バッチまたはダイレクトコマンドのみ）	INPLでは、バイナリやポータブルなど、ロードファイルのタイプが自動的に認識されます。ただし、INPLをバッチモードまたはダイレクトコマンドモードで実行する場合は、互換性を保持するために `File type` パラメータを指定する必要があります。このパラメータは評価されません。
Load File	ロードするファイルの名前。
Report File	**INPL レポート**（次を参照）が書き込まれるファイルの名前。

INPL レポート

選択した INPL 機能が完了すると、対応する INPL レポートが［**Report File**］フィールドで指定したファイルに書き込まれます。ワークファイルを指定しないと、レポートが画面に表示されます。

Load Libraries Only

INPLユーティリティのこの機能は、Naturalのカタログ化オブジェクトやソースオブジェクト、および共有リソースを Natural システムファイル FNAT または FUSER の指定されたライブラリにロードするために使用します。

手順 4.2. ライブラリをロードするには

1 INPL メニューでファンクションコード「`L`」を入力します。この機能の実行時に有効になるパラメータを指定できます。

- **Replace** (Y/N)
- **Load Except** (Y/N)
- ライブラリ
- **Object Name**
- **Date** (YYYY-MM-DD)

- **Check Date** (Y/N)
- **Number**

これらのパラメータの詳細については、「*INPL ユーティリティの概要*」セクションの「*使用可能なオプション*」を参照してください。

2 エントリを確定します。

機能が完了すると、対応する **INPL レポート**（「*INPL ユーティリティの概要*」を参照）が出力されます。

Load DDMs Only

INPL ユーティリティのこの機能は、DDM を ワークファイルで指定されたライブラリにロードするために使用します。

手順 4.3. DDM をロードするには

1 INPL メニューでファンクションコード「`D`」を入力します。この機能の実行時に有効になるパラメータを指定できます。

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

これらのパラメータの詳細については、「*INPL ユーティリティの概要*」セクションの「*使用可能なオプション*」を参照してください。

2 エントリを確定します。

機能が完了すると、対応する **INPL レポート**（「*INPL ユーティリティの概要*」を参照）が出力されます。

Load Error Messages Only

INPL ユーティリティのこの機能は、ユーザー定義のエラーメッセージまたはシステムエラーメッセージをそれぞれ Natural システムファイル FUSER または FNAT にロードするために使用します。

手順 4.4. エラーメッセージをロードするには

1 INPL メニューでファンクションコード「E」を入力します。この機能の実行時に有効になるパラメータを指定できます。

- **Replace** (Y/N)
- ライブラリ

これらのパラメータの詳細については、「*INPL ユーティリティの概要*」セクションの「*使用可能なオプション*」を参照してください。

2 エントリを確定します。

機能が完了すると、対応する **INPL レポート**（「*INPL ユーティリティの概要*」を参照）が出力されます。

Load All Objects

INPL ユーティリティのこの機能は、エラーメッセージと DDM を含むすべての Natural オブジェクトおよび共有リソースをワークファイル 1 で指定されたライブラリにロードするために使用します。

手順 4.5. すべてのオブジェクトおよび共有リソースをロードするには

1 INPL メニューでファンクションコード「B」を入力します。この機能の実行時に有効になるパラメータを指定できます。

- **Replace** (Y/N)
- **Load Except** (Y/N)
- **DDM Name**
- ライブラリ
- **Object Name**
- **Date** (YYYY-MM-DD)
- **Check Date** (Y/N)

- **Number**

これらのパラメータの詳細については、「*INPL ユーティリティの概要*」セクションの「*使用可能なオプション*」を参照してください。

2 エントリを確定します。

機能が完了すると、対応する **INPL レポート**（「*INPL ユーティリティの概要*」を参照）が出力されます。

Scan INPL File

INPL ユーティリティのこの機能は、ワークファイル 1 に割り当てられたファイルの内容をスキャンするために使用します。

手順 4.6. INPL ファイルをスキャンするには

1 INPL メニューでファンクションコード「S」を入力します。この機能の実行時に有効になるパラメータを指定できます。

- **Load Except** (Y/N)
- **DDM Name**
- ライブラリ
- **Object Name**
- **Date** (YYYY-MM-DD)
- **Number**

これらのパラメータの詳細については、「*INPL ユーティリティの概要*」セクションの「*使用可能なオプション*」を参照してください。

2 エントリを確定します。

機能が完了すると、対応する **INPL レポート**（「*INPL ユーティリティの概要*」を参照）が出力されます。

Natural Security Recover

INPL ユーティリティのこの機能は、Natural Security 環境を初期化するために使用します。

次のオプションを使用できます。

- 環境のリセット
- オーナーの削除

環境のリセット

注意: この機能を実行すると、ユーザープロファイル DBA、ライブラリプロファイル SYSSEC、初期インストール後のこの2つのオブジェクト間のリンクがリセットされます。ライブラリ SYSSEC へのそれ以外のリンクはすべてキャンセルされます。他の Natural Security プロファイルとリンクは変更されません。詳細については、Software AG 技術サポートに連絡してください。

手順 4.7. 環境をリセットするには

- INPL メニューでファンクションコード「R」を入力します。

オーナーの削除

手順 4.8. オーナーを削除するには

- 指定したオブジェクトのオーナー情報をリセットするには、INPL メニューでファンクションコード「R」を入力し、［**Replace**］フィールドに「O」を入力します。

ユーザー出口ルーチン

INPL ユーザー出口ルーチンは、Natural システムライブラリ SYSLIB のソースオブジェクト INPLSX*nn* として提供されます（*nn* はユーザー出口ルーチンの ID）。

手順 4.9. ユーザー出口ルーチンを起動するには

1. SYSLIB からユーザーライブラリにソースコードをコピーします。
2. 名前 INPLUX*nn*. の下でカタログ化
3. Natural システムライブラリ SYSLIB にコピーして戻します。

注意: ユーザーが変更した可能性のあるソースオブジェクト、およびユーザー出口ルーチンのカタログ化オブジェクトは、更新インストールによって上書きされないように名前が変更されます。

次のユーザー出口ルーチンを使用することができます。

Name	関数
INPLUX01	エラーメッセージテキストを置き換えを禁止します。

INPLUX01

このユーザー出口を使用して、INPL セッション中に置換できないエラーメッセージ（ユーザー定義または Natural システムエラーメッセージ）の範囲を定義できます。詳細については、Natural システムライブラリ SYSLIB の INPLSX01 のソースを参照してください。

IV Natural Termcap（NATTERMCAP）ユーティリティ

5 Natural Termcap（NATTERMCAP）ユーティリティ

- 全体的な設定 36
- Natural Termcap ユーティリティの呼び出し 38
- Natural Termcap ユーティリティの終了 42
- Terminal Copy Capabilities 43
- キー定義 44
- ［File］メニュー 44
- ［Edit］メニュー 45
- ［Search］メニュー 46
- ［Test］メニュー 46
- ［Options］メニュー 47
- ［Help］メニュー 47
- 端末機能 - 概要 47
- 端末機能 - 名前別 54

Natural Termcap（NATTERMCAP）ユーティリティは、Naturalで使用される端末機能の作成、変更、およびテストを行うために使用します。これらの端末機能は、端末データベースSAGtermcap に格納されています。

標準の端末タイプ定義はないため、Software AG は、SAGtermcap に含まれる端末タイプの完全性と正確性について責任を負いません。Digital Equipment Corporation の VT 端末に使用される端末タイプ標準は、ANSI X3.64（ISO 6429 に対応）です。

全体的な設定

このセクションでは、Natural Termcap ユーティリティの一般的な設定を指定するために使用できる環境変数とパラメータについて説明します。Natural Termcap ユーティリティを呼び出すときに指定できるパラメータについては、「ダイナミックパラメータ」を参照してください。

- 環境変数
- 端末モード
- 特殊な制御コード

環境変数

Natural Termcap ユーティリティでは、次の環境変数が使用されます。

変数	説明
TERM	現在アクティブになっている端末タイプに使用されます。
NATTERM	Natural 端末タイプに使用されます。この変数が設定も定義もされていない場合、TERM（上記参照）に割り当てられている値が使用されます。
NATTCAP	SAGtermcap とは別の端末データベースを指定するために使用されます。
COLUMNS	端末の画面幅に使用されます。この環境変数が設定も定義もされていない場合、現在の画面幅が使用されます。それ以外の場合、機能 co に割り当てられている値が使用されます。
LINES	端末画面のページサイズに使用されます。この環境変数が設定も定義もされていない場合、現在の画面ページサイズが使用されます。それ以外の場合、機能 li に割り当てられている値が使用されます。

端末モード

Natural Termcap ユーティリティは、画面 I/O アプリケーションです。このため、画面にメニューとウィンドウを表示するには、端末データベースと端末タイプが必要です。デフォルトでは、Natural Termcap ユーティリティでは Natural と同じ端末データベースおよびタイプが使用されます。

Natural では、次のようにして端末タイプが取得されます。

1. 環境変数 NATTERM の内容が取得されます。
2. NATTERM が設定も定義もされていない場合、システム環境変数 TERM の内容が取得されます。

Natural では、次のようにして端末データベースが取得されます。

1. 環境変数 NATTCAP の内容が取得されます。
2. NATTCAP が設定されていない場合、ローカルコンフィグレーションファイル *NATURAL.INI* の NATTCAP エントリから端末データベース名が取得されます。
3. この端末データベースが見つからなかった場合、Natural Termcap ユーティリティは、現在のディレクトリ内で SAGtermcap という名前の端末データベースを見つけようとします。

この自動処理が行われないようにするために、Natural Termcap ユーティリティには、端末タイプの数が制限されたダイナミックパラメータ TERMCAP において定義済みの端末データベースが提供されています。これらのタイプにアクセスするには、ダイナミックパラメータ DISPLAY を指定します。

特殊な制御コード

次の表に、印刷できない文字と、端末機能構文において特殊な意味を持つ文字の概要を示します。

制御コード	説明
\E	エスケープ文字。
\b	バックスペース文字。
\n	改行。
\r	改行キー。
\t	タブ文字。
\xxx	xxx の 8 進数値。3 文字でなければなりません。
\072	文字コロン（:）。Natural Termcap ユーティリティでは、コロン（:）は内部セパレータとして使用されます。
^x	Control-x。x は任意の文字です。

例：

DEC VT220 端末でファンクションキー PF10 を F10 として定義する場合、F10 のコードは次のようになります。

```
<ESCAPE>[21~
```

機能 PF10 を次のように指定します。

```
\E[21~
```

キーボードで文字破線（~）を使用できない場合は、代わりに波線の 8 進数値を使用してください。波線は 8 進数 176 として定義されます。この場合は、次のように指定することもできます。

```
\E[21\176
```

Natural Termcap ユーティリティの呼び出し

手順 5.1. Natural Termcap ユーティリティを呼び出すには

■ オペレーティングシステムプロンプトで、次のように入力します。

```
nattermcap [dynamic-parameters]
```

dynamic-parameters は、コマンドで指定できる 1 つ以上のパラメータを示します（「ダイナミックパラメータ」を参照）。

［**NATURAL Termcap Utility**］画面の右上隅に、現在アクティブな端末の名前が表示されます。

この画面には、次のメニューオプションがあります。

Menu	説明
File	端末エントリの作成、読み取り、保存、および削除を行います。
Edit	端末機能を表示および設定します。
Search	特定の機能を名前で検索します。
Test	機能をテストします。
Options	デフォルトのキー定義を変更し、端末コピー機能の表示と非表示を切り替えます。端末コピー機能は、別の端末エントリから取り込まれた機能です。
Help	各機能およびダイナミックパラメータの使用法に関するヘルプを提供します。

次の各セクションで、［**NATURAL Termcap Utility**］画面に表示されるメニューについて説明します。

ダイナミックパラメータ

次のセクションでは、Natural Termcap ユーティリティを呼び出すときに NATTERCAMP コマンドとともに指定できるダイナミックパラメータについて説明します。これらのパラメータを使用すると、端末タイプの機能にすばやくアクセスできます。別の方法として、［**NATURAL Termcap Utility**］画面に表示されるメニューオプションを使用することもできます。

パラメータ構文に使用されている記号については、『*システムコマンド*』ドキュメントの「*システムコマンド構文*」を参照してください。

ダイナミックパラメータ	説明
DISPLAY	Natural Termcap ユーティリティ自体の端末タイプを定義するために使用されます。DISPLAY パラメータが指定されていない場合、Natural 端末データベース SAGtermcap が使用され、Natural と同じ端末タイプ記録メカニズムが有効になります。
	構文： DISPLAY = {#vt100|#vt100ng|#vt220|#vt220ng|wyse60|#tty|*other*}
	設定可能値：

#vt100	内部端末データベースから端末エントリ DEC vt100 を使用します。
#vt100ng	#vt100 と似ていますが、グラフィックライン文字が -（マイナス記号）、|（縦線）、+（プラス記号）などの単一文字で置き換えられます。
#vt220	内部端末データベースから端末エントリ DEC vt220 を使用します。
#vt220ng	#vt220 と似ていますが、グラフィックライン文字が -（マイナス記号）、|（縦線）、+（プラス記号）などの単一文字で置き換えられます。
#wyse60	内部端末データベースから端末エントリ wyse60 を使用します。

ダイナミックパラメータ	説明	
	#tty	内部端末データベースから端末エントリ tty を使用します。tty 端末は、エスケープコントロールシーケンスを使用せず、行単位で動作します。この端末エントリが選択されている場合、数個の機能しか使用できません。
	other	端末データベース内の、例えば xterm などのその他の任意の端末タイプを使用します。
EDIT	特定の端末機能を表示／変更する場合に使用します。この機能が見つかると、関連付けられたウィンドウが表示され、指定されたフィールドにカーソルが置かれます。この機能が見つからない場合、エラーメッセージが表示され、Natural Termcap ユーティリティが終了します。 構文： EDIT = *capability* 機能： Natural で認識される任意の端末機能を指定できます。機能のリストは、「*端末機能 - 概要*」および「*端末機能 - 名前別*」セクションを参照してください。	
EXIT	すべてのパラメータの処理の完了後にユーティリティを終了する場合に使用します。 例： NATTERM EDIT = PF10 EXIT ファンクションキー PF10 を変更した後、ユーティリティはすぐに終了します。	
HELP	特定の機能または Natural Termcap ユーティリティの使用法に関するヘルプを取得する場合に使用します。	
	構文： HELP = {CAP|USAGE|*capability*}	
	設定可能値：	
	CAP	すべての機能のヘルプが機能名別にソートされて表示されます。
	USAGE	Natural Termcap ユーティリティのすべてのダイナミックパラメータが表示されます。

ダイナミックパラメータ	説明
	capability 特定の機能のヘルプが表示されます。
REPORT	Natural TMP ディレクトリ内の現在の端末の詳細な説明を含むテキストファイルをデフォルトで作成する場合に使用します。 構文： REPORT[=*file-name*] *file-name* は、説明を含めるテキストファイルの名前です。 名前を指定しない場合、*terminal-name*.txt が使用されます。
SAVE	現在の端末エントリのすべての変更内容を保存する場合に使用します。 構文： SAVE[=*terminal-name*] *terminal-name*は現在の端末のすべての機能が格納される端末データベース内の新しい端末エントリです。対応するメニューオプションは、［**File**］>［**Save As**］です。 名前を指定しない場合、現在の *terminal-name*が使用されます。対応するメニューオプションは、［**File**］>［**Save**］です。
TERM	別の端末エントリ内を読み取る場合に使用します。このパラメータが指定されていない場合、現在の Natural 端末タイプが使用されます（NATTERM または TERM）。 構文： TERM = *terminal-name* *terminal-name* は、端末データベースに含まれる任意の端末タイプです。
TERMCAP	別の端末データベースを使用する場合に使用します。このパラメータが指定されていない場合、現在の Natural 端末データベースが使用されます（NATTCAP）。 構文： TERMCAP = *database-name* *database-name* は、データベースパスとファイル名です。
TEST	指定したテストウィンドウが開きます。 構文： TEST = {CONSISTENCY|COLORS|GRAPHICS|KEYS|VIDEO}

ダイナミックパラメータ	説明	
	設定可能値：	
	CONSISTENCY	ファンクションキーが一意に定義されているかどうかがチェックされます。
	COLORS	すべての使用可能な色がサンプルテキストとともに表示されます。
	GRAPHICS	一重線および二重線のボックスが表示されます。
	KEYS	押されたキーのテキストが表示されます。ウィンドウを閉じるには、次の文字キーを押します。E、Q、X、または .（ピリオド）
	VIDEO	点滅、下線、反転表示などのビデオ属性が表示されます。

Natural Termcap ユーティリティの終了

手順 5.2. Natural Termcap ユーティリティを終了するには

■ ［**File**］メニューで、［**Exit**］を選択します。

または:

EXIT パラメータを設定します（「ダイナミックパラメータ」を参照）。

Natural Termcap ユーティリティが終了し、オペレーティングシステムプロンプトが表示されます。

Terminal Copy Capabilities

端末コピー機能（TC）は、Cプログラムの`#include`命令のように、別の端末エントリから転送された機能です。ただし、現在のエントリ内で機能がすでに定義されている場合、転送された機能は無視されます。これにより、冗長性が減り、関連するエントリの一貫性が保たれるため、エントリの効率が上がります。端末コピーエントリから読み取られた機能は、入力フィールドの右側に`[TC]`というマークが付きます。また、この機能の転送元のエントリの名前が、メニュー右上隅にある端末名の上に表示されます。

端末機能が変更されると、転送済みの端末エントリへのリンクは失われ、現在の端末エントリが変更されます。

手順 5.3. 端末コピー機能なしで現在の端末エントリを表示するには

1 ［**NATURAL Termcap Utility**］画面の［**Options**］メニューで［**Terminal Copy Capabilities**］を選択します。

   ［**Terminal Copy Capabilities**］ウィンドウが表示されます。

2 **HIDE** を選択します。

例：

`TERM`が`vt100`に設定されている場合、端末データベース内の vt100（vt220）端末エントリは次のようになります。

vt100 エントリ：

```
ti = \E =
ESC = \E
ETO = 300
tc = vt220
```

vt220 エントリ：

```
ti = \E[0m
cr = \r
```

端末タイプ vt100 の結合された端末エントリは次のようになります。

vt100：

```
ti = \E = /* taken from the original vt100 entry
ESC = \E  /* taken from the original vt100 entry
ETO = 300 /* taken from the original vt100 entry
cr = \r   /* transferred from vt220 entry
```

`ti` はすでに vt100 エントリ内に定義されているため、vt220 端末からの機能 `ti( = \E[0m)` は無視されます。

キー定義

次の表で、Natural Termcap ユーティリティの定義ウィンドウで使用できるファンクションキーについて説明します。

キー	説明
CTRL+A	ANSI 定義を挿入します。
CTRL+E	自動的にキーを評価します。
CTRL+N	フレームの非グラフィック文字を挿入します。
CTRL+P	ヘルプを取得します。
CTRL+V	機能をテストします。

手順 5.4. 事前定義されたファンクションキーを変更するには

1 ［**NATURAL Termcap Utility**］画面の［**Options**］メニューで［**Key Assignments**］を選択します。

［**Key Assignments**］ウィンドウが表示されます。

2 ［**Key Name**］列の下にリストされた必要なファンクションキーを変更します。

指定できるのは Ctrl キー（CTRL+A～CTRL+Z）のみです。

［File］メニュー

［**NATURAL Termcap Utility**］画面から［**File**］を選択すると、次の機能を含む選択リストが表示されます。

関数	説明
New	現在の端末データベース内に新規の端末エントリを作成します。
Read	端末データベースから端末エントリを読み取ります。
Save	端末機能を現在の端末エントリに保存します。
Save As	端末機能を別の端末エントリまたは新規の端末エントリに保存します。
Delete	現在の端末エントリを端末データベースから削除します。
Generate Report	現在の端末エントリの説明、エイリアス名、および機能の情報を含むテキストファイルを生成します。デフォルトでは、テキストファイルは Natural TMP ディレクトリに *terminal-name*.txt（例えば xterm.txt）として保存されます。
Move	端末エントリを端末データベースの先頭に物理的に移動します。端末がデータベースの先頭にある場合、端末初期化中のアクセス時間が向上します。
Import Database	Natural の SAGtermcap 以外の端末データベースを使用できます。
Export Database	端末データベース全体およびすべての端末エントリを、Natural の SAGtermcap とは別のパス／名前で保存します。
Properties	端末データベース、端末エントリ、環境変数、および表示タイプの詳細情報を表示します。
Exit	Natural Termcap ユーティリティを終了します。

［Edit］メニュー

［**NATURAL Termcap Utility**］画面から［**Edit**］を選択すると、すべての機能がトピック別に分類された選択リストが表示されます。

- **Colors**
- **Cursor Keys and Modes**
- **Description and Comments**
- **Editing Key**
- **Initialization and Reset**
- **Keypad Keys for Mathematical Operations**
- **Line Graphics**
- **Miscellaneous**
- **Name and Aliases**
- **PA and PF Keys**
- **Right-To-Left Support**
- **Screen Dimension and Appearance**
- **Video Attributes**

トピックを選択した後、対応するウィンドウが表示され、個々のNatural端末機能を編集できます。

表示されたウィンドウに応じて、次の各種の入力フィールドを使用できます。

- ブール。ON または OFF のみを指定できます。
- 数値。0～9 の数字のみを指定できます。
- 文字列。32 文字の英数字を指定できます。ただし、端末機能が te および ti の場合は、132 文字を指定できます。
- 説明。132 文字の英数字を指定できます。

編集対象の各 Natural 端末機能の詳細については、「*端末機能 - 概要*」および「*端末機能 - 名前別*」セクションを参照してください。

［Search］メニュー

［**NATURAL Termcap Utility**］画面から［**Search**］を選択すると、すべての機能が名前別にソートされた選択リストが表示されます（「*端末機能- 名前別*」を参照）。機能を選択すると、ウィンドウが表示されます。このウィンドウは、該当する［**Edit**］メニューオプションで呼び出されるウィンドウに対応しています。

［Test］メニュー

［**NATURAL Termcap Utility**］画面から［**Test**］を選択すると、次の機能を含む選択リストが表示されます。

関数	説明
Colors	すべての前景および背景をテストします。
Consistency	矛盾するキー定義を検索します。定義が一意でない場合、影響を受けるキーのリストが表示されます。
Keys	押されたキーの名前を表示します。この機能を終了するには、次のいずれかの文字キーを押します。E、Q、X、または .（ピリオド）
Line Graphics	ウィンドウフレームの描画に使用されるグラフィックライン機能をテストします。
Video Attributes	すべてのビデオ属性をテストします。

［Options］メニュー

［**NATURAL Termcap Utility**］画面から［**Options**］を選択すると、次の機能を含む選択リストが表示されます。

関数	説明
Terminal Copy Capabilities	機能 tc で指定された別の端末エントリから取り込まれた端末機能の表示と非表示を切り替えます。
Key Assignments	ユーティリティのデフォルトのキー割り当てを変更します。

［Help］メニュー

［**NATURAL Termcap Utility**］画面から［**Help**］を選択すると、次の機能を含む選択リストが表示されます。

関数	説明
Topics	特定のトピック（NAME、REPORT、TC または TEST など）の詳細なヘルプセクションを呼び出します。
Capabilities	各機能の詳細なヘルプセクションを呼び出します。
Usage	ダイナミックパラメータの使用法に関する情報を表示します。
About	製品情報を表示します。

端末機能 - 概要

このセクションでは、すべての端末機能の概要について、トピックでソートして説明します。これらのトピックは、［**Edit**］メニューの項目に対応しています。

- Colors
- カーソルキーおよびモード
- 説明およびコメント
- キーの編集
- 初期化およびリセット
- 算術演算用のキーパッドキー
- 線のグラフィック
- その他
- 名前およびエイリアス名
- PA キーおよび PF キー
- Right-To-Left サポート

- 画面の次元および外観
- ビデオ属性

Colors

Name	説明
ct	端末タイプ：色（ON）またはモノクロ（OFF）
bgbla	画面の背景色
fgblu	前景は青、Natural カラー定義 CD=BL（*）
fggre	前景は緑、Natural カラー定義 CD=GR（*）
fgmag	前景はピンク、Natural カラー定義 CD=PI（*）
fgred	前景は赤、Natural カラー定義 CD=RE（*）
fgcya	前景は青緑、Natural カラー定義 CD=TU（*）
fgwhi	前景は白、Natural カラー定義 CD=NE（*）
fgyel	前景は黄、Natural カラー定義 CD=YE（*）
ctres	反転表示用の前景、Natural 属性定義 AD=V（*）
bgblu	背景色は青
bggre	背景色は緑
bgmag	背景色はピンク
bgred	背景色は赤
bgcya	背景色は青緑
bgwhi	背景色は白
bgyel	背景色は黄色

* Natural 定義 AD および CD の詳細については、*Natural リファレンス*ドキュメントに記載されている該当するセッションパラメータ AD および CD を参照してください。

カーソルキーおよびモード

カーソルキーは、アプリケーションモードと通常（数値）モードの 2 つのモードで設定できます。アプリケーションモードでは、数値キーパッドキーが通常モードとは異なるタスクに割り当てられています。

Name	説明
kd	下方向カーソルキー（標準モード）
kl	左方向カーソルキー（標準モード）
kr	右方向カーソルキー（標準モード）
ku	上方向カーソルキー（標準モード）
@7	End カーソルキー

Name	説明
kh	Home カーソルキー
CKNO	標準カーソルキーモード
CKAP	アプリケーションカーソルキーモード
cm	カーソル動作
CNL	カーソル改行
ve	カーソル表示
vi	カーソル非表示
DK	下方向カーソルキー（アプリケーションモード）
LK	左方向カーソルキー（アプリケーションモード）
RK	右方向カーソルキー（アプリケーションモード）
UK	上方向カーソルキー（アプリケーションモード）

説明およびコメント

これらのフィールドを使用すると、端末エントリに関する説明を記述すること、およびコメントを追加することができます。

キーの編集

Name	説明
bc	バックスペースキー
bcvt	代替バックスペースキー
bt	バックタブキー
cr	改行キー
dc	文字削除キー
KDEL	フィールド末尾まで削除キー
ESC	エスケープキー
ETO	エスケープのタイムアウト値（ミリ秒単位）
%1	ヘルプキー
kI	挿入または上書きモードキー
NLFF	最初のフィールドの次の行へキー
kN	ページダウン（次へ）キー
PD	代替ページダウン（次へ）キー
kP	ページアップ（前へ）キー
PU	代替ページアップ（前へ）キー
&2	リフレッシュキー
ta	タブキー

初期化およびリセット

Name	説明
TICL	初期化後に画面をクリア
TIRA	初期化後に属性をリセット
TIGR	初期化後に線のグラフィックを有効化
TICI	初期化後にカーソル非表示
TICV	初期化後にカーソル表示
TIAK	初期化後にアプリケーションキーパッド
TINK	初期化後に数値キーパッド
TIAC	初期化後にアプリケーションカーソルキーモード
TINC	初期化後に標準カーソルキーモード
TIDB	初期化後に暗い背景
TILB	初期化後に明るい背景
TIIM	初期化後に挿入モード
TIOM	初期化後に上書きモード
ti	追加の初期化シーケンス
TECL	終了後に画面をクリア
TERA	終了後にビデオ属性をリセット
TENL	終了後にカーソル改行
TECV	終了後にカーソル表示
TECI	終了後にカーソル非表示
TEAK	終了後にアプリケーションキーパッド
TENK	終了後に数値キーパッド
TEAC	終了後にアプリケーションカーソルキーモード
TENC	終了後に標準カーソルキーモード
TEDB	終了後に暗い背景
TELB	終了後に明るい背景
te	終了後の追加のシーケンス

算術演算用のキーパッドキー

キーパッド	説明
KP01	単一の空値
KP1	1
KP2	2
KP3	3
KP4	4
KP5	5
KP6	6
KP7	7
KP8	8
KP9	9
KPADD	加算
KPSUB	減算
KPDIV	除算
KPMUL	乗算
KPTS	千の位での区切り
KPDP	小数点
KPRES	Result

線のグラフィック

Name	説明
eA	線のグラフィックを有効化
as	グラフィックモードをオン
ae	グラフィックモードをオフ
G1	右上隅の 1 バイト文字
GD1	右上隅の 2 バイト文字
G2	左上隅の 1 バイト文字
GD2	左上隅の 2 バイト文字
G3	左下隅の 1 バイト文字
GD3	左下隅の 2 バイト文字
G4	右下隅の 1 バイト文字
GD4	右下隅の 2 バイト文字
GH	水平バーの 1 バイト文字
GDH	水平バーの 2 バイト文字

Name	説明
GV	垂直バーの 1 バイト文字
GDV	垂直バーの 2 バイト文字

その他

Name	説明
bl	アラーム
vb	アラーム表示
cl	画面のクリア
ce	行末までクリア
ks	アプリケーションキーパッドモード
ke	数値キーパッドモード
xi	スクロール異常
TCS	外部の端末／プリンタ文字セット（詳細は『*オペレーション*』ドキュメントの「*NATCONV.INI での異なる文字セットのサポート*」を参照）
tc	端末コピー

名前およびエイリアス名

各エントリに対して、1 個の名前および最大 30 個のエイリアス名を定義できます。

PA キーおよび PF キー

キー	説明
PA1	アテンションキー PA1
PA2	アテンションキー PA2
PA3	アテンションキー PA3
PF1	ファンクションキー PF1
PF2	ファンクションキー PF2
...	
PF47	ファンクションキー PF47
PF48	ファンクションキー PF48

Right-To-Left サポート

Name	説明
RTLF	右書き言語用フィールドトグルキー
RTLS	右書き言語用画面トグルキー
RTLM	RTL 入力モードの設定
LTRM	LTR 入力モードの設定

画面の次元および外観

Name	説明
li	画面の行の数（指定されていない場合、現在の画面サイズが使用される）
co	画面の列の数（指定されていない場合、現在の画面サイズが使用される）
DAR	暗い背景、明るいテキスト
LIG	明るい背景、暗いテキスト

ビデオ属性

Name	説明
mb	点滅オン、Natural 属性定義 `AD=B`（*）
BR	点滅オフ
adc	手書き／斜体オン、Natural 属性定義 `AD=C`（*）
adc0	手書き／斜体オフ
md	高輝度（太字）オン、Natural 属性定義 `AD=I`（*）
HR	高輝度オフ
mr	反転オン、Natural 属性定義 `AD=V`（*）
mr0	反転オフ
us	下線オン、Natural 属性定義 `AD=U`（*）
ue	下線オフ
me	属性のリセット、Natural 属性定義 `AD=D`（*）
so	強調表示モードオン
se	強調表示モードオフ
xs	強調表示異常

* Natural 属性定義 `AD` の詳細については、『*Natural リファレンス*』ドキュメントに記載されている該当するセッションパラメータ `AD` を参照してください。

端末機能 - 名前別

このセクションでは、すべての端末機能を名前別にソートして示します。これらの機能は、［**Search**］メニューの機能に対応しています。

Name	説明
%1	ヘルプキー
&2	リフレッシュキー
@7	End カーソルキー
adc	手書き／斜体オン、Natural 属性定義 AD=C（*）
adc0	手書き／斜体オフ
ae	グラフィックモードをオフ
as	グラフィックモードをオン
bc	バックスペースキー
bcvt	代替バックスペースキー
bgbla	画面背景
bgblu	背景色は青
bgcya	背景色は青緑
bggre	背景色は緑
bgmag	背景色はピンク
bgred	背景色は赤
bgwhi	背景色は白
bgyel	背景色は黄色
bl	アラーム
BR	点滅オフ
bt	バックタブキー
ce	行末までクリア
CKAP	アプリケーションカーソルキーモード
CKNO	標準カーソルキーモード
cl	画面のクリア
cm	カーソル動作
CNL	カーソル改行
co	画面の列の数（指定されていない場合、現在の画面サイズが使用される）
cr	改行キー
ct	カラー端末
ctres	反転表示用の前景、Natural 属性定義 AD=V（*）

Name	説明
DAR	暗い背景、明るいテキスト
dc	文字削除キー
DK	下方向カーソルキー（アプリケーションモード）
eA	線のグラフィックを有効化
ESC	エスケープキー
ETO	エスケープのタイムアウト値
fgblu	前景は青、Natural カラー定義 CD=BL（*）
fgcya	前景は青緑、Natural カラー定義 CD=TU（*）
fggre	前景は緑、Natural カラー定義 CD=GR（*）
fgmag	前景はピンク、Natural カラー定義 CD=PI（*）
fgred	前景は赤、Natural カラー定義 CD=RE（*）
fgwhi	前景は白、Natural カラー定義 CD=NE（*）
fgyel	前景は黄、Natural カラー定義 CD=YE（*）
G1	右上隅の 1 バイト文字
G2	左上隅の 1 バイト文字
G3	左下隅の 1 バイト文字
G4	右下隅の 1 バイト文字
GD1	右上隅の 2 バイト文字
GD2	左上隅の 2 バイト文字
GD3	左下隅の 2 バイト文字
GD4	右下隅の 2 バイト文字
GDH	水平バーの 2 バイト文字
GDV	垂直バーの 2 バイト文字
GH	水平バーの 1 バイト文字
GV	垂直バーの 1 バイト文字
HR	高輝度オフ
kd	下方向カーソルキー（標準モード）
KDEL	フィールド末尾まで削除キー
ke	数値キーパッドモード
kh	Home カーソルキー
kI	挿入または上書きモードキー
kl	左方向カーソルキー（標準モード）
kN	ページダウン（次へ）キー
kP	ページアップ（前へ）キー
KP01	単一ゼロ（0）キーパッドキー
KP1	1

Name	説明
KP2	2
KP3	3
KP4	4
KP5	5
KP6	6
KP7	7
KP8	8
KP9	9
KPADD	加算
KPDIV	除算
KPDP	小数点
KPMUL	乗算
KPRES	Result
KPSUB	減算
KPTS	千の位での区切り
kr	右方向カーソルキー（標準モード）
ks	アプリケーションキーパッドモード
ku	上方向カーソルキー（標準モード）
li	画面の行の数（指定されていない場合、現在の画面サイズが使用される）
LIG	明るい背景、暗いテキスト
LK	左方向カーソルキー（アプリケーションモード）
LTRM	LTR 入力モードの設定
mb	点滅オン、Natural 属性定義 AD=B（*）
md	高輝度（太字）オン、Natural 属性定義 AD=I（*）
me	属性のリセット、Natural 属性定義 AD=D（*）
mr	反転オン、Natural 属性定義 AD=V（*）
mr0	反転オフ
NLFF	最初のフィールドの次の行へキー
PA1	アテンションキー PA1
PA2	アテンションキー PA2
PA3	アテンションキー PA3
PD	代替ページダウン（次へ）キー
PF1	ファンクションキー PF1
PF2	ファンクションキー PF2
...	
PF47	ファンクションキー PF47

Name	説明
PF48	ファンクションキー PF48
PU	代替ページアップ（前へ）キー
RK	右方向カーソルキー（アプリケーションモード）
RTLF	右書き言語用フィールドトグルキー
RTLM	RTL 入力モードの設定
RTLS	右書き言語用画面トグルキー
se	強調表示モードオフ
so	強調表示モードオン
ta	タブキー
tc	端末コピー
TCS	外部の端末／プリンタ文字セット
te	終了後の追加のシーケンス
TEAC	終了後にアプリケーションカーソルキーモード
TEAK	終了後にアプリケーションキーパッド
TECI	終了後にカーソル非表示
TECL	終了後に画面をクリア
TECV	終了後にカーソル表示
TEDB	終了後に暗い背景
TELB	終了後に明るい背景
TENC	終了後に標準カーソルキーモード
TENK	終了後に数値キーパッド
TENL	終了後にカーソル改行
TERA	終了後にビデオ属性をリセット
ti	追加の初期化シーケンス
TIAC	初期化後にアプリケーションカーソルキーモード
TIAK	初期化後にアプリケーションキーパッド
TICI	初期化後にカーソル非表示
TICL	初期化後に画面をクリア
TICV	初期化後にカーソル表示
TIDB	初期化後に暗い背景
TIGR	初期化後に線のグラフィックを有効化
TIIM	初期化後に挿入モード
TILB	初期化後に明るい背景
TINC	初期化後に標準カーソルキーモード
TINK	初期化後に数値キーパッド
TIOM	初期化後に上書きモード

Name	説明
TIRA	初期化後に属性をリセット
ue	下線オフ
UK	上方向カーソルキー（アプリケーションモード）
us	下線オン、Natural 属性定義 AD=U（*）
vb	アラーム表示
ve	カーソル表示
vi	カーソル非表示
xi	スクロール異常
xs	強調表示異常

* Natural 定義 AD および CD の詳細については、*Natural リファレンス*ドキュメントに記載されている該当するセッションパラメータ AD および CD を参照してください。

V オブジェクトハンドラ

オブジェクトハンドラは、Natural 環境での分散のために Natural および Natural 以外のオブジェクトを処理するように設計されています。これは、ソース環境のオブジェクトをワークファイルにアンロードし、それらをワークファイルからターゲット環境にロードすることによって実行されます。

「*オブジェクトハンドラ*」ドキュメントは、次の内容で構成されています。

オブジェクトハンドラの全般的な情報	バッチまたはオンラインモードでのオブジェクトハンドラの呼び出し、Natural Security の適用。
機能	オブジェクトハンドラメニュー機能（アンロード、ロード、ロードの再スタート、スキャン、表示、検索、および管理）の使用。
オブジェクト指定	オブジェクトハンドラメニュー機能で処理するオブジェクトの指定：**Natural** ライブラリオブジェクト、**Natural** システムエラーメッセージ、**Natural** コマンドプロセッサ、外部オブジェクト および **FDT**
設定	オブジェクトハンドラメニュー機能のためのオプションとパラメータ設定の指定。
「ワークプラン」	オブジェクトハンドラ機能を実行するための標準プロシージャの使用。
名前と日時の指定	名前、日付、時刻、および範囲の指定。
ワークファイル	オブジェクトハンドラに使用されるワークファイル。
ダイレクトコマンド	オブジェクトハンドラ機能を実行するためのダイレクトコマンドの使用。
バッチのコンディションコードとユーザー出口ルーチン	バッチモードで提供されるコンディションコードとユーザー出口ルーチン。
Tools	ステータス情報の表示およびトレースとレポートオプションの設定。
プロファイル設定	個々のデフォルトプロシージャおよび標準プロシージャを定義するためのプロファイルの設定。
NATUNLD/NATLOAD および **SYSTRANS** からオブジェクトハンドラへの移行	ユーティリティ NATUNLD/NATLOAD および SYSTRANS からオブジェクトハンドラへの移行

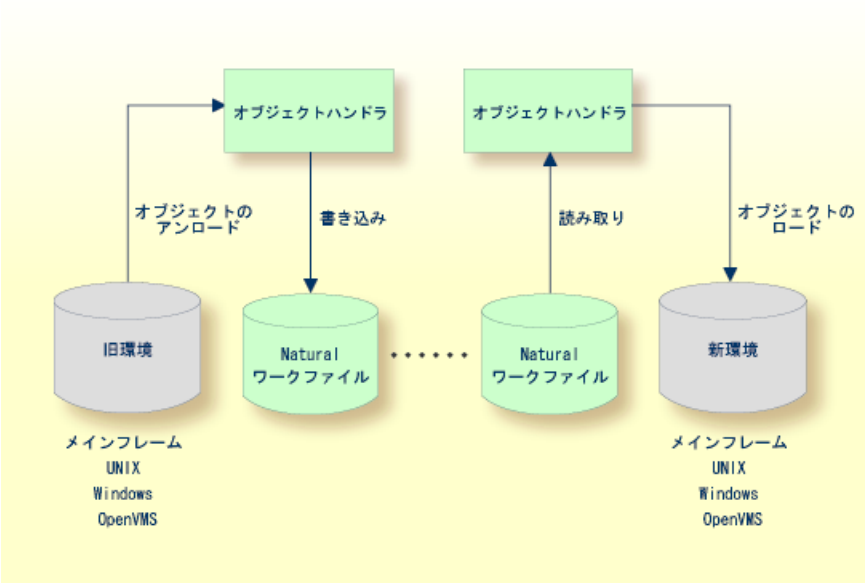
6 オブジェクトハンドラの全般的な情報

- オブジェクト転送の原理 62
- オブジェクトハンドラの呼び出し 64
- バッチまたはダイレクトコマンドの呼び出し 65
- Natural プログラムからのオブジェクトハンドラコマンドの発行 66
- Natural Security 66
- 標準の PF キー 67
- FDDM システムファイルの使用 67

オブジェクトハンドラは、Natural システムライブラリ SYSOBJH 内にあるユーティリティ SYSOBJH、およびダイレクトコマンドインターフェイスで構成されています。また、Natural プログラムからオブジェクトハンドラ機能を実行するために、アプリケーションプログラミングインターフェイス OBJHAPI が用意されています。

オブジェクト転送の原理

下の図に、オブジェクトをソース環境からワークファイルにアンロードし、ワークファイルからターゲット環境にロードすることによって、オブジェクトがオブジェクトハンドラでどのように転送されるかを示します。必要に応じて、ソース環境からターゲット環境へのワークファイルの転送に、FTP などのアプリケーションプロトコルを使用できます。



このセクションでは、次のトピックについて説明します。

- 転送環境とファイルセキュリティ
- オブジェクトハンドラによって処理されるオブジェクト

- フォーマットオプション

転送環境とファイルセキュリティ

新旧の環境は、Adabas データベースに含まれている FNAT、FUSER、および FDIC の各システムファイル、またはメインフレーム上のVSAMファイルシステム、またはUNIX、OpenVMS、およびWindowsプラットフォーム上のファイルシステムです。次のセクションで説明するように、FNAT または FUSER システムファイルの Natural オブジェクトはライブラリに含めることができます。

ファイルセキュリティ（パスワードとサイファコード）は、Adabas または VSAM 環境のシステムファイルに対して定義されたセキュリティと関連付けられています。システムファイルにセキュリティが定義されている場合、オブジェクトハンドラ機能を実行する前に、必要なソース／ターゲットシステムファイルに対して、パスワード、サイファコード、およびVSAM 名を指定する必要があります。これを行わなかった場合、AdabasまたはVSAMによって該当するエラーメッセージが発行されます。オブジェクトハンドラの開始時にNaturalセッションに割り当てられているデフォルトのシステムファイルについては、セキュリティ情報を指定する必要はありません。

オブジェクトハンドラによって処理されるオブジェクト

オブジェクトハンドラは、Natural ライブラリに含まれている Natural ソースオブジェクト（保存オブジェクトとも呼ばれる）およびカタログ化オブジェクト、Natural エラーメッセージ、Natural コマンドプロセッサソース、Natural 関連オブジェクト、Adabas FDT（フィールド定義テーブル）および外部ファイル（外部オブジェクト）を転送します。

フォーマットオプション

オブジェクトが処理されるソース環境およびターゲット環境に応じて、バイナリフォーマットまたはテキストフォーマットでデータを転送できます。

バイナリフォーマットは、ソースオブジェクトおよびカタログ化オブジェクト、エラーメッセージ、Natural コマンドプロセッサソース、Natural 関連オブジェクト、Adabas FDT、および外部ファイル（外部オブジェクト）に使用できます。

テキストフォーマットは、ソースオブジェクト、Natural コマンドプロセッサソース、エラーメッセージ、および Adabas FDT に適用されます。メインフレームおよび UNIX/OpenVMS/Windows プラットフォーム間では、テキストデータのみを転送できます。同一プラットフォーム間では、バイナリデータを転送できます。UNIX または OpenVMS および Windows プラットフォーム間では、内部フォーマットのポータブルワークファイルを使用してバイナリデータを転送できます。

オブジェクトハンドラの呼び出し

オブジェクトハンドラを呼び出すには、メニュー機能またはダイレクトコマンドを使用できます。

手順 6.1. 任意の Natural ライブラリからオンラインでオブジェクトハンドラを呼び出すには

1 次のシステムコマンドを入力します。

```
SYSOBJH
```

入力モードに応じて、オブジェクトハンドラ **Main Menu** またはオブジェクトハンドラ **Compact Menu** のいずれかが表示されます。「*コンパクトモード*」を参照してください。入力モードを変更するには、`SET INPUT` コマンドを使用するか（「*ダイレクトコマンド*」を参照）、または `INPUT-MODE` パラメータを変更します（「*プロファイルの設定*」を参照）。オブジェクトハンドラの **Main Menu** では次のオプションが提供されます。

- アンロード
- ロード
- スキャン
- ビュー
- 管理

これらの機能の詳細と、上級ユーザーモードまたはウィザードでこれらの機能を実行する方法については、「*機能*」セクションを参照してください。

2 次のいずれかの方法で機能を選択します。

目的の機能に対応する項目の横にある入力フィールドに任意の文字を入力します。

または:

目的の機能に対応する PF キーを押します。

または:

コマンド行で、目的の機能に対応するオブジェクトハンドラコマンドを入力します。有効なコマンドの詳細については、「*ダイレクトコマンド*」セクションを参照してください。

手順 6.2. バッチモードまたはダイレクトコマンドオンラインモードでオブジェクトハンドラを呼び出すには

- 「*バッチまたはダイレクトコマンドの呼び出し*」および「*ダイレクトコマンド*」の説明に従って、システムコマンド `SYSOBJH` の後にダイレクトコマンドを入力します。

ダイレクトコマンドの実行後は、別のダイレクトコマンドを入力するか、ピリオド（.）を入力してオブジェクトハンドラを終了します。

バッチまたはダイレクトコマンドの呼び出し

複数のコマンドをオンラインモードまたはバッチモードでオブジェクトハンドラに発行できます。コマンドシーケンスの最後のコマンドは、ピリオド（.）、STOP、END、QUIT または FIN にする必要があります。FIN にすると Natural セッションが終了します。

このセクションでは、次のトピックについて説明します。

- バッチモード
- オンラインモード

バッチモード

オブジェクトハンドラへのコマンドは標準入力から読み込まれます。各コマンドは、キーワードまたはキーワードの後の入力デリミタ（セッションパラメータ ID）を入力することによって最大 20 のコマンドパーツ／文字列に分けることができます。各コマンドパーツ／文字列は 248 バイト以内でなければなりません。

コマンドが1行より長い場合、コマンドに属している各行（最後の行を除く）の末尾にセッションパラメータ CF で定義された文字（デフォルトは %）を入力します。これは次の行への継続を示します。ただし、これは、コマンド SYSOBJH を単独で 1 行に指定する場合に限り可能です。つまり、複数行コマンドが開始する同じ行に SYSOBJH を入力した場合は CF を使用できません。また、LS プロファイルパラメータを 250 に設定することをお勧めします。

例（**ID** が , に設定されているとみなす）：

```
UNLOAD * LIB EXAMPLE, WHERE, WORK $HOME\TEST.SAG
   STOP
```

関連トピック：

- *ダイレクトコマンド*
- 『*オペレーション*』ドキュメントの「*バッチモードでの Natural*」

オンラインモード

コマンド行に入力されるオブジェクトハンドラへのコマンドは、最高 20 のコマンドパーツで構成できます。

例：

```
SYSOBJH UNLOAD * LIB EXAMPLE WHERE TRANSFER WORK $HOME/TEST.DAT
```

Natural プログラムからのオブジェクトハンドラコマンドの発行

Natural システムライブラリ SYSOBJH 内にサブプログラムとして提供されている OBJHAPI アプリケーションプログラミングインターフェイスを使用して、Natural プログラムからオブジェクトハンドラにコマンドを発行できます。必要なパラメータおよび例については、ライブラリ SYSOBJH にある Natural プログラム DOC-API を参照してください。

Natural Security

Natural Security 環境でオブジェクトハンドラを使用するには、オブジェクトハンドラに対してユーティリティプロファイルを Natural Security で定義する必要があります。少なくとも、デフォルトのプロファイルを定義する必要があります。ユーティリティプロファイルの詳細については、『*Natural Security*』ドキュメントの「*Protecting Utilities*」セクションを参照してください。

Natural Security がインストールされている場合、オブジェクトハンドラは、要求された機能が有効かどうかおよびパラメータ設定が許可されているかどうかを確認するために、Natural Security の SYSOBJH ユーティリティプロファイルをチェックします。

ロード機能の実行中に Natural Security エラーが発生すると、次の操作が行われます。

- ［**Write report**］オプションがオンラインモードで設定されていると、エラーメッセージがレポートファイルに出力され、現在のロードコマンドの処理は続行します。
- ［**Write report**］オプションがバッチモードで設定されていると、エラーメッセージがレポートファイルに出力され、エラーが発生したロードコマンドの処理が完了した後、オブジェクトハンドラは終了します。
- ［**Write report**］オプションが設定されていないと、エラーメッセージが発行され、ロードコマンドは終了します。

標準の PF キー

次の PF キーは、すべてのフルスクリーンマップで使用できます。

PF キー	説明
PF1	カーソルが置かれているフィールドのヘルプ機能を呼び出します。 フィールド［**Work file**］、［**External path**］、［**Ext. Path**］、［**Object name**］、**Report file**］または［**Restart file**］にカーソルを移動すると、最大 253 文字の長いワークファイル名を入力できる別のウィンドウが表示されます。
PF3	現在の画面を終了し、前の画面に戻ります。
PF6	リストの先頭にジャンプします。
PF7	リストを 1 ページ分スクロールアップします。 画面の場合は、1 画面／ステップ戻ります。
PF8	リストを 1 ページ下方にスクロールします。 ウィザード画面の場合は、次の画面／ステップに進みます。
PF9	リストの最後にジャンプします。
PF10	ナビゲーションおよび特別な設定を行うためのコマンドを選択する［**Commands**］メニューが表示されます。「*ダイレクトコマンド*」の「*ナビゲーションと特別な機能のためのコマンド*」も参照してください。
PF12	現在の機能をキャンセルします。
PF20	オブジェクトハンドラのアクティブなプログラムをすべてリスト表示します。技術的な問題を Software AG に連絡する際に役立つ情報です。

FDDM システムファイルの使用

Natural DDM（データ定義モジュール）は、ライブラリまたはシステムファイル FDDM に保存できます。以下の項目も参照してください。「*FDDM - DDM 用の Natural システムファイル*」（『*パラメータリファレンス*』ドキュメント）

ロード、アンロード、または検索機能で DDM の処理にシステムファイル FDDM を使用するために、オブジェクトハンドラには、［**Use FDDM file for processing DDMs**］オプションが用意されています。このオプションは、**Set additional options** を使用して設定します（「*設定*」セクションを参照）。

Use FDDM file for processing DDMs を選択する際は、次の点に注意してください。

- FDDM が NATPARM パラメータファイルでアクティブになっていると、このオプションがデフォルトで選択されます。

- ライブラリに保存されている DDM は処理できません。
- ライブラリ SYSTEM および Natural オブジェクトタイプ V を指定する必要があります（「*オブジェクト指定*」セクションの「***Natural** ライブラリオブジェクトの詳細*」を参照）。
- ロード機能で使用すると、すべての DDM が システムファイル FDDM にロードされます。この場合、パラメータ NEWLIBRARY は無視されます。

7 機能

このセクションでは、オブジェクトハンドラが提供する主な機能について説明します。

オブジェクトハンドラウィザードを利用して、アンロード、ロード、およびスキャン機能の実行に必要な手順を段階的に進めていくことができます。ウィザードはデフォルトでアクティブになります。代わりに上級ユーザー用のアンロード、ロード、またはスキャンモードを使用する場合は、**Main Menu** の［**Advanced user**］の横にあるフィールドを選択します。さらに、上級ユーザーの場合は、内部コマンド `SET INPUT-MODE C` を入力して、アンロード、ロード、またはスキャン機能にコンパクト入力モードを使用できます。また、適切なオブジェクトハンドラのプロファイルオプション **Input-Mode** を使用して、デフォルトの入力モード（ウィザード、上級、またはコンパクト）を設定することもできます。「*プロファイルの設定*」セクションも参照してください。

ヒント: 繰り返し設定とオブジェクト指定を定義する標準プロシージャを作成して、アンロード、ロード、またはスキャン機能の処理を自動化することができます。「ワークプラン」を参照してください。

このセクションでは、次のトピックについて説明します。

ウィザード
上級ユーザー
コンパクトモード
ロードの再スタート
ビュー
検索
管理
ワークプランライブラリの変更
システムファイルの選択
ライブラリの選択
ワークプランのリストと選択

システムエラーメッセージの選択
オブジェクトの選択

注意:

1. 「*ワークプランライブラリの変更*」については、「*管理*」セクションで説明しています。
2. 「*ワークプランのリストと選択*」については、「*管理*」セクションの「*ワークプランライブラリ内の利用可能なワークプランのリストの作成*」で説明しています。

8 ウィザード

- 手順 1 - プロシージャの開始 72
- 手順 2 - ワークファイルへのオブジェクトのアンロード／ワークファイルからのオブジェクトのロード／ワークファイルのオブジェクトのスキャン 73
- 手順 3 - パラメータの設定 75
- 手順 4 - オブジェクトの選択 76
- 手順 5 - 処理の実行 78
- 手順 6 - 処理の続行 78

オブジェクトハンドラでは、ウィザードを使用して次の処理の実行順序を指定できます。

- Natural システム環境から Natural ワークファイルへのデータのアンロード。
- ワークファイルから Natural システム環境へのデータのロード。
- Natural ワークファイルの内容のスキャン。

手順 8.1. ウィザードをアクティブにするには

- 必要に応じて、**Main Menu** で［**Advanced user**］フィールドを選択します（このフィールドはデフォルトでは未選択）。

画面（手順）間を移動するためのキー PF8 キーおよび PF7 キーがウィザードに表示されます。処理シーケンスをキャンセルするには、PF12 キーを使用します。

このセクションでは、アンロード、ロード、またはスキャンウィザードで実行される処理シーケンスについて説明します。

手順 1 - プロシージャの開始

手順 8.2. アンロード、ロード、またはスキャンプロシージャを開始するには

1 **Main Menu** で、目的の機能の横に任意の単一文字を入力するか、または対応する PF キーを使用して、［**Unload**］、［**Load**］、または［**Scan**］を選択します。

次のオプションを含む最初の **Wizard** 画面が表示されます。

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

2 オブジェクトをワークファイルにアンロードする場合、オブジェクトをワークファイルからロードする場合、またはワークファイルのオブジェクトをスキャンする場合は、次の「*手順 2 - ワークファイルへのオブジェクトのアンロード／ワークファイルからのオブジェクトのロード／ワークファイルのオブジェクトのスキャン*」に進みます。

または:

オブジェクトのアンロード、ロード、またはスキャン用のコマンドプロシージャを使用する場合は、［**Start Object Handler command procedure**］を選択して次のように進みます。

1. 最初の［**Wizard**］画面で［**Start Object Handler command procedure**］を選択します。［**Procedure**］画面が表示されます。
2. 次のいずれかのオプションを使用して、タイプ PROCEDURE のワークプラン名を［**Name**］フィールドに入力します。

- トランザクションで使用するタイプ PROCEDURE のワークプラン名を入力します（「ワークプラン」も参照）。
- ［**Select Workplan**］を選択するか、または PF5 キーを押して、タイプ PROCEDURE の使用可能なワークプランのリストを表示します。選択するワークプランの横にある行で、コマンド「S」または「SE」を入力します。ENTER キーを押してコマンドを実行し、［**Procedure**］画面の［**Name**］フィールドに値を入力します。

3. 指定したワークプランを表示するには、［**List Workplan**］を選択するか、または PF4 キーを押します。

   「*管理*」セクションの「*ワークプランライブラリ内の利用可能なワークプランのリストの作成*」も参照してください。

3 ENTER キーを押して続行します。

4 「*手順 5 - 処理の実行*」に進みます。

手順 2 - ワークファイルへのオブジェクトのアンロード／ワークファイルからのオブジェクトのロード／ワークファイルのオブジェクトのスキャン

手順 8.3. Natural ワークファイルからオブジェクトをアンロードする／Natural ワークファイルにオブジェクトをロードする／Natural ワークファイルをスキャンするには

1 最初の［**Wizard**］画面で［**Unload/Load/Scan objects into/from Natural work file(s)**］を選択します。

2 ENTER キーを押すか PF8 キー（次へ）を押して続行します。ウィザードの［**Options**］画面が開きます。この画面では、次のフィールド、コマンド、および代替用の PF キーを使用できます。

フィールド	PF キー	説明
Transfer format		有効になるのは、［**Use default options**］（これがデフォルトです）を選択した場合に限られます。 これを選択すると、ワークファイルにおける処理データの書き込み／読み取りが転送フォーマットで実行されます。「ワークファイル」の「ワークファイルフォーマット」も参照してください。 アンロード機能： アンロードするデータは、転送フォーマットでワークファイルに書き込まれます。以降のアンロードではこのフィールドの設定を変更する場合は、［**Main Menu**］に戻るか、コマンド GO UNLOAD END（「ダイレクトコマンド」の「ナ

フィールド	PF キー	説明
		ビゲーションと特別な機能のためのコマンド」を参照）を入力してアンロード機能を再スタートする必要があります。 ロード／スキャン機能： ロードまたはスキャンするデータは転送フォーマットにする必要があります。
Unicode work file		アンロード機能にのみ適用され、有効になるのは［**Transfer format**］が選択されている場合に限られます。 このオプションが選択されている場合、すべてのオブジェクトソースが、ワークファイルに書き込まれる前に Unicode/UTF-8（Universal Transformation Format、8 ビット形式）に変換されます。
Use PC File		適用されるのは、Entire Connection がインストールされている場合に限られます。 これを選択すると、処理データが Entire Connection ワークファイルから読み込まれるか、このワークファイルに書き込まれます。
Portable work file		ロード／スキャン機能を使用する場合には設定する必要はありません。これらの機能では自動的に適切なワークファイルタイプが選択され、このオプションは設定されていても無視されます。 ［**Portable work file**］が有効になるのは、次の条件に該当した場合に限られます。 ■ ［**Use default options**］（これがデフォルトです）を選択した場合。 ■ ［**Transfer format**］が選択されていない場合。 ［**Portable work file**］を選択すると、ワークファイルの書き込み／読み取りはポータブルフォーマットで実行されます。「ワークファイル」の「*ワークファイルフォーマット*」も参照してください。
Work file		有効になるのは、**Use default options**（これがデフォルトです）を選択した場合に限られます。 ■ その機能を使用するワークファイルの名前。名前が使用可能なスペースを超える場合、PF11 キー（WorkF）を押すと名前をさらに長く入力できます（最大 253 文字）。または、このフィールドでカーソルを表示して PF1（ヘルプ）キーを押します。 「*ワークファイル*」を参照してください。 ■ または、**Use PC File** が選択されている場合、パス名および Entire Connection ワークファイルが使用されます。 「*ワークファイル*」を参照してください。
Use default options		デフォルトオプションが使用されます（これがデフォルトです）。使用可能なオプションについては、「*プロファイルの設定*」および「*設定*」の「*その他のオプションの設定*」を参照してください。

フィールド	PF キー	説明
Set additional options	PF4	有効になるのは、［**Use default options**］を選択した場合に限られます。 ウィザードの［**Options**］画面を表示します。この画面で、デフォルト設定を変更したり、処理の順序を指定する追加オプションを入力したりすることができます。「設定」の「*その他のオプションの設定*」を参照してください。
Use Option Workplan		これが選択されている場合、タイプ OPTION のワークプランが使用されます（「*ワークプラン*」を参照）。
Name		有効になるのは、［**Use Option Workplan**］を選択した場合に限られます。 使用するタイプ OPTION のワークプランの名前。
List Option Workplan	PF6	有効になるのは、［**Use Option Workplan**］を選択した場合に限られます。 ［Name］フィールドに指定されたワークプランの内容を表示します。
Select Option Workplan	PF5	有効になるのは、［**Use Option Workplan**］を選択した場合に限られます。 タイプ OPTION の使用可能なワークプランの選択リストが表示されます（「*管理*」の「*ワークプランライブラリ内の利用可能なワークプランのリストの作成*」も参照）。

3 表示されたオプションを選択し、処理順序の指定に使用されるフィールドに（必要に応じて）値を入力します。

4 ENTER キーを押すか PF8 キー（次へ）を押して続行します。

ウィザードの［**Parameters**］画面が表示されます。

手順 3 - パラメータの設定

手順 8.4. 処理プロシージャのパラメータを設定するには

1 ［**Parameters**］画面で次のオプションを選択し、（必要に応じて）処理シーケンスに使用されるフィールドに値を入力します。

フィールド	PF キー	説明
Do not use parameters		これが選択されている場合（デフォルト設定）、パラメータは設定されません。
Use global parameters		これが選択されている場合、グローバルパラメータが使用されます。「設定」の「*グローバルパラメータの設定*」を参照してください。

フィールド	PF キー	説明
Set global parameters	PF4	有効になるのは、［**Use global parameters**］を選択した場合に限られます。 これが選択されている場合、［**Parameters**］画面が表示されます。キーワードと有効な入力値の詳細については、「*設定*」の「*グローバルパラメータの設定*」および *parameter-setting*（ダイレクトコマンド）に関する項を参照してください。
Use Parameter Workplan		これが選択されている場合、タイプ PARAMETER のワークプランが使用されます（「*ワークプラン*」を参照）。
Name		有効になるのは、［**Use Parameter Workplan**］を選択した場合に限られます。 使用するタイプ PARAMETER のワークプランの名前。
List Parameter Workplan	PF6	有効になるのは、［**Use Parameter Workplan**］を選択した場合に限られます。 これが選択されている場合、［**Name**］フィールドで指定したワークプランの内容が表示されます。
Select Parameter Workplan	PF5	有効になるのは、フィールド［**Use Parameter Workplan**］を選択した場合に限られます。 これが選択されている場合、タイプ PARAMETER の使用可能なワークプランの選択リストが表示されます（「*管理*」の「*ワークプランライブラリ内の利用可能なワークプランのリストの作成*」を参照）。

2 ENTER キーを押すか PF8 キー（次へ）を押して続行します。

［**Select Unload/Load/Scan Type**］画面が表示されます。

手順 4 - オブジェクトの選択

手順 8.5. 処理するオブジェクトのタイプを選択するには

1 ［**Select Unload/Load/Scan Type**］画面で、以下の 3 つのオプションのいずれかを選択します。最初のオプションは、ロードおよびスキャン機能にのみ適用されることに注意してください。各オブジェクトタイプに適用されるキーワードと有効な値については、「*オブジェクト指定*」セクションの該当する記載を参照してください。

1. ワークファイルのすべてのオブジェクトを処理するには、［**Load/Scan all objects**］を選択します。
2. 特定タイプのオブジェクトを選択します。
   - **Natural** ライブラリオブジェクト

- **Natural** システムエラーメッセージ
- **Natural** コマンドプロセッサソース
- **Natural** 関連オブジェクト
- 外部オブジェクト
- **FDT**

ENTER キーを押すか PF8 キー（次へ）を押して続行します。

選択したオブジェクトのタイプに応じて画面が表示され、処理するオブジェクトの選択条件を指定できます。

選択条件を入力し、必要に応じて［**Details**］を選択し（使用可能な場合）オブジェクトの詳細指定を行います。［**Details**］の詳細については、「*オブジェクト指定*」セクションの関連する説明を参照してください。

3. タイプ SELECTION または LIST のワークプランを使用するには、［**Use Selection or List Workplan**］を選択します。「*ワークプラン*」も参照してください。

   ENTER キーを押すか PF8 キー（次へ）を押して続行します。

   ［**Selection or List**］画面が表示されます。次のいずれかのオプションを使用して、タイプ SELECTION または LIST のワークプラン名を［**Name**］フィールドに入力します。

   - ワークプランの名前を入力します。

     または：

     ［**Select Workplan**］を選択するか、または PF5 キー（SelWP）を押して、使用可能なすべてのワークプランのリストを表示します。選択するワークプランの横にある行で、コマンド「S」または「SE」のいずれかを入力します。

     ENTER キーを押してコマンドを実行し、［**Selection or List**］画面の［**Name**］フィールドに値を入力します。「*管理*」セクションの「*ワークプランライブラリ内の利用可能なワークプランのリストの作成*」も参照してください。

   ［**Name**］フィールドに入力したワークプランの内容を表示するには、［**List Workplan**］を選択するか、または PF4 キー（Li-WP）を押します。

2 ENTER キーを押すか PF8 キー（次へ）を押して続行します。

入力データから生成された処理コマンドがウィザードに表示されます。

コマンド SAVE を入力するか PF5 キー（保存）を押すと、タイプ PROCEDURE のワークプランとして表示されるコマンドを保存できます（「*ワークプラン*」も参照）。

手順 5 - 処理の実行

手順 8.6. 処理プロシージャを実行するには

1 コマンド実行画面で ENTER キーを押すか、または PF8 キー（次へ）を押して設定を確認し、指定したオブジェクトを処理します。

必要に応じて PF7 キー（戻る）を押して、コマンドの実行を確認する前に処理設定を修正します。

オブジェクトハンドラで機能が実行され、確認メッセージが表示されます。

2 ENTER キーを押して続行します。

処理されたオブジェクトのリストを示すレポート画面が表示されます。

3 PF3 キー（Exit）を押してレポート画面を終了するか、または PF12 キー（Canc）を押して機能を終了します。

ウィンドウが表示され、データの処理を続行するかどうかを選択できます。

4 ［**No**］を選択してから ENTER キーを押して機能を終了します。

または:

PF12 キーを押して機能を終了します。

［**Main Menu**］が表示されます。

手順 6 - 処理の続行

手順 8.7. 処理を続行するには

1 レポート画面で PF3 キー（Exit）を押します。

ウィンドウが表示され、次の処理手順に進むかどうかを選択できます。

2 ［**Yes**］を選択します。

前の設定を再利用または変更するオプションを示す画面が表示されます。

9 上級ユーザー

- 上級ユーザーのアクティブ化 80
- オブジェクトの処理 80

このセクションでは、上級ユーザーモードを呼び出す方法、およびアンロード、ロード、スキャンの各機能を実行する方法について説明します。

注意: このパラメータは現在使用されていません。互換性維持の目的でのみ保持されています。

上級ユーザーのアクティブ化

手順 9.1. 上級ユーザーモードをアクティブにするには

- **Main Menu** で［**Advanced user**］フィールドを選択します（このフィールドはデフォルトでは未選択）。

  または:

  オブジェクトハンドラプロファイルの `Advanced-Mode` パラメータを指定して、上級ユーザーモードをデフォルトとして設定します（「*プロファイルの設定*」を参照）。

オブジェクトの処理

注意: FDT をロードする場合は、「*オブジェクト指定*」セクションの「***FDT***」も参照してください。

手順 9.2. 上級ユーザーモードでオブジェクトを処理するには

1. **Main Menu** で［**Advanced user**］フィールドを確認し、［**Unload**］、［**Load**］または［**Scan**］を選択します。
2. ENTER キーを押して続行します。

   セクション［**Options**］および［**Parameters**］が表示された［**Unload/Load/Scan Settings**］画面が開きます。
3. 「*設定*」セクションの説明に従って、オプションおよびパラメータを設定します。
4. ENTER キーを押して続行します。

   ［**Select Unload/Load/Scan Type**］画面が表示されます。
5. 処理するオブジェクトを選択します。「*オブジェクト指定*」セクションも参照してください。
6. ［**Details**］を選択して、追加の選択条件を指定します。「*オブジェクト指定*」の関連するセクションを参照してください。

7 ENTER キーを押して続行します。

- オブジェクトハンドラプロファイルでパラメータ `Display-Cmd-in-Advanced-Mode` が `N`（No）に設定されている場合（これがデフォルトです）、またはそのようなプロファイルが存在しない場合は、入力データから生成されたコマンドが、選択データを指定した直後に実行されます。「*プロファイルの設定*」も参照してください。

  ［**Write report**］フィールド（これがデフォルトです）を選択した場合、［**Display Unload/Load/Scan Report**］画面が開き、処理されたオブジェクトのリストが表示されます。「*設定*」セクションの「*ワークファイルオプション*」も参照してください。

- オブジェクトハンドラプロファイルでパラメータ `Display-Cmd-in-Advanced-Mode` が `Y`（Yes）に設定されている場合（「*プロファイルの設定*」を参照）、またはコマンド `SET ADVANCEDCMD ON`（「*ナビゲーションと特別な機能のためのコマンド*」を参照）が前に実行されている場合は、入力データから生成されたコマンドを示す画面が表示されます。

  コマンド `SAVE` を入力するか PF5 キー（保存）を押すと、タイプ PROCEDURE のワークプランとして表示されるコマンドを保存できます（「*ワークプラン*」も参照）。

  ENTER キーを押してコマンドの実行を確認するか、または PF3 キー（Exit）を押してコマンドの実行を確認する前に処理設定を変更します。

  ［**Write report**］フィールド（これがデフォルトです）を選択した場合、［**Display Unload/Load/Scan Report**］画面が開き、処理されたオブジェクトのリストが表示されます。「*設定*」セクションの「*ワークファイルオプション*」も参照してください。

10 コンパクトモード

- コンパクトモードの選択方法 84
- コンパクトモードでの命令の処理のされ方 84

コンパクトモードを選択すると、上級ユーザーはアンロード、ロード、またはスキャン機能を実行するオブジェクトパラメータをわずか 2 手順で指定できます。他のオブジェクトパラメータも、適切なサブメニューを選択することで指定または変更することができます。コンパクトモードのもう1つの機能は、「ウィザードモード」（「*ウィザード*」を参照）に記載されている処理手順を、手順のシーケンス全体をパスすることなく実行することです。したがって、コンパクトモードは、上級モードよりも強力であり、さらなる専門知識が必要です。この章では、次のトピックについて説明します。

コンパクトモードの選択方法

コンパクトモードを選択するには、以下の 2 つの方法があります。

- オブジェクトハンドラプロファイルを調整する
  「*プロファイル設定*」で INPUT-MODE パラメータに関する説明を参照してください。
- コマンド SET INPUT-MODE C またはSET IM C
  「*ダイレクトコマンド*」の「*ナビゲーションと特別な機能のためのコマンド*」で SET INPUT コマンドに関する説明を参照してください。

コンパクトモードでの命令の処理のされ方

コンパクトモードでは、処理指示は2つの手順で実行できます。手順1では、コンパクトモードメインメニューから実行する機能を選択します。その結果、選択した機能に固有の 2 つ目のメニューが表示されます。ここには、手順1で選択した機能に固有のパラメータを入力できます。このセクションには、次の両方のメニューが含まれます。

- コンパクトモードメインメニュー
- コンパクトモード後のメニュー

コンパクトモードメインメニュー

コンパクトモードメインメニューでは、次のフィールドを指定できます。

フィールド	PF キー	説明
Function		機能を選択します。 Uアンロード（デフォルト） Lロード Sスキャン

フィールド	PF キー	説明
		注意: アンロード機能を開始すると、メニューを終了してコンパクトモードを再起動し、アンロード機能を終了しない限り、**Function** の変更は許可されません。
Object		処理するオブジェクトのタイプ： LNatural ライブラリオブジェクト（デフォルト） ENatural システムエラーメッセージ CNatural コマンドプロセッサソース RNatural 関連オブジェクト X外部オブジェクト FFDT 任意の Natural オブジェクト。有効になるのは、**Function.** に対して［L］または［S ］を選択した場合に限られます。
Work file format		処理するデータに適用するロードおよびアンロード操作のファイルフォーマット： I内部フォーマット（デフォルト） T転送フォーマット 「ワークファイルフォーマット」を参照してください。
Portable work file		ロード／スキャン機能を使用する場合には設定する必要はありません。これらの機能では自動的に適切なワークファイルタイプが選択され、このオプションは設定されていても無視されます。 ［**Portable work file**］が有効になるのは、次の条件に該当した場合に限られます。 ■ ［**Use default options**］（これがデフォルトです）を選択した場合。 ■ ［**Transfer format**］が選択されていない場合。 ［**Portable work file**］を選択すると、ワークファイルの書き込み／読み取りはポータブルフォーマットで実行されます。「ワークファイル」の「ワークファイルフォーマット」も参照してください。
Use PC file		適用されるのは、Entire Connection がインストールされている場合に限られます。 これを選択すると、処理データが Entire Connection ワークファイルから読み込まれるか、このワークファイルに書き込まれます。
PC file		適用されるのは、Entire Connection がインストールされている場合に限られます。 Entire Connection ワークファイルへの完全パス名。現在のシステム環境でバックスラッシュ（\）セパレータを使用できない場合は、代わりにスラッシュ（/）を使用します。

フィールド	PF キー	説明
Write report		「Y」に設定されている場合、処理するオブジェクトのレポートを［**Report text member**］フィールドで指定したレポートテキストオブジェクトに書き込みます。［**Write report**］オプションはデフォルトで「Y」に設定されています。 レポートを表示するには、内部コマンド SHOW REPORT FILE を入力します（「*ダイレクトコマンド*」の「*ナビゲーションと特別な機能のためのコマンド*」を参照）。
Report text member		有効になるのは、［**Write report**］を選択した場合に限られます。レポートが書き込まれる、ワークプランライブラリに保存されたテキストオブジェクトの名前。
Set additional options	PF4	「"Y"」に設定した場合、処理シーケンスのデフォルト設定を修正し、追加オプションを入力できる［**Options**］画面が表示されます。使用可能なオプションについては、「*その他のオプションの設定*」を参照してください。**Use Option Workplan** を「Y」に設定した場合は使用できないか、オプション指定が無視されます。注：いずれかのオプションが定義されている場合は、テキスト「（オプションが定義されています）」が表示されます。
Use Option Workplan		「Y」に設定されている場合は、タイプ OPTION のワークプランが使用されます。「*ワークプラン*」も参照してください
Option Workplan name		使用するタイプ OPTION のワークプランの名前。
	PF5	タイプ OPTION の使用可能なワークプランの選択リストが表示されます。「*管理*」の「*ワークプランライブラリ内の利用可能なワークプランのリストの作成*」も参照してください。
	PF6	［**Option Workplan name**］フィールドに指定されたワークプランの内容を表示します。
Set global parameters	PF7	［**Parameters**］画面が表示されます。キーワードと有効な入力値の詳細については、「*グローバルパラメータの設定*」および「***parameter-setting***」（「*ダイレクトコマンド*」）を参照してください。 **Use Parameter Workplan** が「Y」に設定されている場合は使用できないか、パラメータ指定が無視されます。 注：いずれかのパラメータが定義されている場合は、テキスト「（Parameters are defined）」が表示されます。
Use Parameter Workplan		「Y」に設定されている場合、タイプ PARAMETER のワークプランが使用されます。「*ワークプラン*」も参照してください。
Parameter Workplan name		使用する PARAMETER ワークプランの名前。
	PF8	タイプ PARAMETER の使用可能なワークプランの選択リストが表示されます。「*管理*」の「*ワークプランライブラリ内の利用可能なワークプランのリストの作成*」も参照してください。
	PF9	**Parameter Workplan name** で指定したワークプランの内容が表示されます。

コンパクトモード後のメニュー

コンパクトモード後のメニューでは、コンパクトモードのメインメニューで行った選択に固有の情報を追加します。指定する属性に関するドキュメントについては、「*オブジェクト指定*」を参照してください。

11 ロードの再スタート

ロードの再スタート機能を使用すると、異常終了したロード機能を再スタートさせることができます。ワークファイルの処理が完了する前にロード機能が停止した場合、ロードの再スタート機能を使用して、停止した場所から処理を続行できます。

ロードの再スタートでは、ロードに関して指定された選択条件、オプション、およびパラメータ設定に従って、再スタート情報がワークファイル6または指定の再スタートファイルに書き込まれている必要があります。

手順 11.1. ロード中に環境を設定するには

1 ［**Load Options**］画面：

- ［**Write restart information**］オプションにマークを付けます。
- ［］［**Restart file**］フィールドに、再スタート情報が書き込まれているファイルの名前を入力します。

［**Load Options**］画面については、「*設定*」セクションの「*ワークファイルオプション／レポートオプション*」を参照してください。

2 ロード機能を実行します。

手順 11.2. ロードの中断後にロードの再スタートを実行するには

- オブジェクトハンドラ画面で、コマンド行に次のコマンドを入力します。

```
GO RESTART
```

［**Restart Options**］画面が表示され、［］［**Restart file**］フィールドに名前を入力して、ファイルを指定できます。

または:

次のダイレクトコマンドを使用します。

```
RESTART
```

RESTART の構文については、「*基本的なコマンド構文*」セクションを参照してください。

関連トピック：

「*管理*」セクションの「*ワークプランライブラリの変更*」
「*ナビゲーションと特別な機能のためのコマンド*」セクションの GO RESTART。

12 ビュー

- Natural ライブラリオブジェクト 92
- Natural システムエラーメッセージ 93
- Natural コマンドプロセッサソース 94
- FDT 95

この機能は、Naturalシステム環境に含まれているすべてのオブジェクトを表示するために使用します。選択するオブジェクトのタイプによっては、この機能を使用して必要に応じてオブジェクトを削除することもできます。

手順 12.1. ビュー機能を起動するには

■ **Main Menu** で［**View**］を選択します。

または:

他の任意のオブジェクトハンドラ画面で、次のダイレクトコマンドを入力します。

```
GO VIEW
```

（「*ダイレクトコマンド*」セクションの「*ナビゲーションと特別な機能のためのコマンド*」も参照してください。）

選択可能なすべてのオブジェクトタイプを示す［**Select View Type**］画面が表示されます。

このセクションでは、［**Select View Type**］画面にリストされるオブジェクトタイプの表示方法について説明します。

Natural ライブラリオブジェクト

Naturalライブラリオブジェクトとは、プログラミングオブジェクトおよびユーザー定義エラーメッセージのことです。

手順 12.2. Natural ライブラリオブジェクトを表示するには

1 ［**Select View Type**］画面で［**Natural library objects**］を選択します。

現在のNatural環境で使用可能なすべてのシステムファイルのリストを示す［**View System Files**］画面が表示されます。

画面の列の詳細については、「*システムファイルの選択*」で説明している［**Select System File**］画面を参照してください。列の内容が同じです。

2 ［**Cmd**］列で、選択するシステムファイルの横に任意の単一文字を入力します。デフォルトでは、現在の FUSER または FNAT システムファイルが選択されます。

指定したシステムファイルで使用可能なすべてのライブラリのリストを示す［**View Libraries**］画面が表示されます。

特定のライブラリからライブラリリストを開始したり、［**Library**］フィールドにライブラリ名またはライブラリ名の範囲を入力してオブジェクトにフィルタをかけたりすることがで

きます。有効な名前の範囲については、「*名前と日時の指定*」セクションの「*名前*」を参照してください。

3 ［**Cmd**］列で、選択するライブラリの横に次の行コマンドのいずれかを入力します。

```
L
```

```
LI
```

```
S
```

```
SE
```

指定したライブラリに含まれているすべてのオブジェクトのリストを示す［**View Library Objects**］画面が表示されます。

この画面の詳細については、「*オブジェクトの選択*」で説明している［**List**］画面に関する説明を参照してください。列の内容が同じです。

4 ［**Cmd**］列で、表示するオブジェクトの横に次の行コマンドのいずれかを入力します。

```
L
```

```
LI
```

または:

必要に応じて、削除するオブジェクトの横に次の行コマンドを入力します。

```
DE
```

入力したコマンドに応じて、選択したオブジェクトのソースコードが画面に表示されるか、または削除機能の実行に使用する確認ウィンドウが表示されます。

Natural システムエラーメッセージ

手順 12.3. Natural システムエラーメッセージを表示するには

1 ［**Select View Type**］画面で［**Natural system error messages**］を選択します。

Natural で使用可能なすべてのシステムエラーメッセージのリストを示す［**View System Error Messages**］画面が表示されます。

この画面の詳細については、［**List System Error Messages**］画面に関する説明を参照してください。列の内容が同じです。

2 ［**Cmd**］列で表示するエラーメッセージの横に、次のいずれかの行コマンドを入力します。

```
L
```

```
LI
```

または:

［**Cmd**］列で、目的のオブジェクトの横に次の行コマンドを入力すると、エラーメッセージを削除できます。

```
DE
```

入力したコマンドに応じて、選択したエラーメッセージのソースコードが画面に表示されるか、または削除機能の実行に使用する確認ウィンドウが表示されます。

Natural コマンドプロセッサソース

手順 12.4. Adabas ファイルに保存されている Natural コマンドプロセッサソースを表示するには

1 ［**Select View Type**］画面で［**Natural command processor sources**］を選択します。

［**View Natural Command Processors**］画面が表示されます。

2 目的の Natural コマンドプロセッサソースが現在の FUSER システムファイル（デフォルトで LFILE 190 が設定済み。「*SYSNCP ユーティリティ*」ドキュメントも参照）に保存されていない場合は、［**DBID**］フィールドに必要なデータベース ID を入力し、［**FNR**］フィールドにファイル番号を入力します。

必要に応じて、［**Password**］フィールドに Adabas パスワードを入力し、［**Cipher**］フィールドにサイファコードを入力します。

Natural コマンドプロセッサソースが保存されているすべてのライブラリのリストを示す［**View Libraries**］画面が表示されます。

3 特定のライブラリからライブラリリストを開始したり、［**Library**］フィールドにライブラリ名またはライブラリ名の範囲を入力して Natural コマンドプロセッサソースにフィルタをかけたりすることができます。有効な名前の範囲については、「*名前と日時の指定*」セクションの「*名前*」を参照してください。

4 ［**Cmd**］列で、選択するライブラリの横に次の行コマンドのいずれかを入力します。

```
L
```

```
LI
```

```
S
```

```
SE
```

指定したライブラリに含まれているすべての Natural コマンドプロセッサソースのリストを示す［**View Command Processors**］画面が表示されます。

この画面の詳細については、［**List**］画面に関する説明を参照してください。列の内容が同じです。

5 ［**Cmd**］列で、目的のオブジェクトの横に次の行コマンドを入力すると、そのオブジェクトを削除することができます。

```
DE
```

削除機能の実行に使用する確認ウィンドウが表示されます。

FDT

手順 12.5. Adabas データベースで使用可能な FDT を表示するには

1 ［**Select View Type**］画面で［**FDTs**］を選択します。

［**View FDTs**］画面が表示されます。

2 目的のオブジェクトが現在の FNAT または FUSER システムファイルに保存されていない場合は、［**DBID**］フィールドのデータベース ID を置き換え、必要に応じて［**FNR from**］および［**FNR to**］フィールドに入力されているファイル番号の範囲を置き換えます。

指定したデータベースのファイル範囲にあるすべての FDT のリストを示す［**View FDTs for DBID**］画面が表示されます。

13 検索

この機能は、Natural環境でオブジェクトを検索し、検出されたオブジェクトのレポートリストを生成するために使用します。

手順 13.1. 検索機能を起動するには

- オブジェクトハンドラの任意の画面で、コマンド行に次のコマンドを入力します。

```
GO FIND
```

検索機能で生成されたレポート画面に表示される列の詳細については、「*オブジェクト指定*」セクションを参照してください。`GO FIND`で提供されるサブコマンドについては、「*ダイレクトコマンド*」セクションの「*ナビゲーションと特別な機能のためのコマンド*」を参照してください。

14 管理

- ワークプランライブラリ内の利用可能なワークプランのリストの作成 100
- 新しいワークプランの作成 102
- ワークプランライブラリの変更 104

この機能は、オブジェクトハンドラのワークプランをメンテナンスするために使用します。

ワークプランと適用される構文の詳細については、「*ワークプラン*」セクションおよび「*ダイレクトコマンド*」セクションを参照してください。

このセクションでは、［**Administration**］画面に表示されるオプションについて説明します。ワークプランを変更する手順については、「*ワークプランライブラリ内の利用可能なワークプランのリストの作成*」を参照してください。

ワークプランライブラリ内の利用可能なワークプランのリストの作成

この機能は、ワークプランの編集や実行などの処理を行うため、ワークプランライブラリに含まれているすべてのワークプランをリストし、ワークプランを選択するために使用します。

手順 14.1. ワークプランをリストするには

- ［**Administration**］画面で［**List the available Workplans in the Workplan library**］を選択するか、PF4 キー（リスト）を選択します。

  ワークプランライブラリに含まれているすべてのワークプランのリストを示す［**List Workplans**］画面が表示されます。

  テキストタイプの Natural オブジェクトがワークプランの場合、ワークプランタイプおよびワークプランの説明の最初の 50 バイトが表示されます。PF5 キーを押して、追加情報を表示できます。

［**List Workplans**］画面は、［**Unload/Load/Scan Settings**］画面などに含まれている選択機能によっても表示されます。

［**List Workplans**］画面に表示される列およびワークプランで実行できるコマンドについては、次のセクションで説明します。

- ［List Workplans］画面の列およびコマンド

［List Workplans］画面の列およびコマンド

次の表では、［**List Workplans**］画面に表示される列およびコマンドについて説明します。

各列見出しの下にある入力フィールドを使用して、特定のワークプランからリストを開始したり、ワークプランにフィルタをかけたりすることができます。有効な入力値については、以下の表を参照してください。

列	PF キー	説明	
Cmd		目的のワークプランの横にある入力フィールドで、次の行コマンドを入力できます。	
		C または CH	構文をチェックします。 タイプ PROCEDURE、SELECTION、PARAMETER、および OPTION のワークプランにのみ適用されます。
		DE	ワークプランを削除します。
		ED	ワークプランを編集します。 「*ワークプランの保存*」で説明している［**Save Workplan**］ウィンドウで、ワークプランの名前または説明を変更できます。
		EX	ワークプランを実行します。タイプ PROCEDURE のワークプランにのみ適用されます。
		L または LI	ワークプランをリストします。
		S または SE	現在の機能に使用するワークプランを選択します。 ［**Unload/Load/Scan Settings**］画面などの選択機能から［**List Workplans**］画面を表示した場合にのみ適用されます。
Name		ワークプランの名前。 「*名前と日時の指定*」の「*名前*」の説明に従って、単一の名前または名前の範囲を入力できます。	
Type		PROCEDURE などのワークプランタイプ。 有効な入力値は次のとおりです。	
		PROCEDURE または P	
		SELECTION または S	
		LIST または L	
		PARAMETER または A	
		OPTION または O	
		TEXT または T	

列	PF キー	説明
		また、すべてのタイプとしてアスタリスク（*）を入力したり、SLなどのようにショートタイプの組み合わせを入力したりすることもできます。
Description		ワークプランの説明。 「*名前と日時の指定*」の「*名前*」の説明に従って、単一の説明または説明の範囲を入力できます。
User ID		PF5 キーでのみ表示できます。 ワークプランを作成したユーザーの ID。 「*名前*」の説明に従って、単一のユーザー ID またはユーザー ID の範囲を入力できます。
Date		PF5 キーでのみ表示できます。 ワークプランの作成日。 「*名前と日時の指定*」の「*日付*」の説明に従って、単一の日付または日付の範囲を入力できます。
Time		PF5 キーでのみ表示できます。 ワークプランの作成時刻。 「*名前と日時の指定*」の「*時刻*」の説明に従って、単一の時刻または時刻の範囲を入力できます。
	PF4	追加情報表示（PF5 キー）から標準表示に切り替えます。
	PF5	追加情報（**User ID**、**Date**、および **Time**）を表示します。

新しいワークプランの作成

この機能によって［**Create a new Workplan**］画面が表示され、この画面で、新しいワークプランのタイプおよびワークプランの編集に使用するフォーマットを指定できます。

［**Free Format Editing**］オプションを選択しない（フィールドはマークされず、これがデフォルト設定です）と、タイプ OPTION、PARAMETER、および SELECTION のワークプランでは画面に入力フィールドが表示されます。

［**Free Format Editing**］オプションを選択する（フィールドがマークされる）か、または別のタイプのワークプランを作成すると、ワークプランの内容を入力できる編集エリアを含むマップが表示されます。「ワークプラン」セクションの「*ワークプランの内容*」も参照してください。

フリーフォーマット編集のオン／オフの設定に使用できる代替のダイレクトコマンドについては、「*ダイレクトコマンド*」セクションの「*ナビゲーションと特別な機能のためのコマンド*」にある SET コマンドを参照してください。

このセクションでは、次のトピックについて説明します。

- PROCEDURE ワークプランの作成
- LIST ワークプランの作成
- ワークプランの保存

PROCEDURE ワークプランの作成

現在のオブジェクトハンドラ機能のために生成されたコマンドから、タイプ PROCEDURE のワークプランを作成できます。

手順 14.2. 生成されたコマンドから PROCEDURE ワークプランを作成するには

1 実行する機能用にコマンドが生成されて画面に表示されるまでは、オブジェクトハンドラウィザードを使用して、ワークプランに使用する機能を実行します。

または:

上級ユーザーモードでは、次のいずれかの方法を選択して、生成されたコマンドの表示をアクティブにします。

- 次のオブジェクトハンドラコマンドを入力します。

```
SET ADVANCEDCMD ON
```

または：
オブジェクトハンドラプロファイルで、パラメータ Display-Cmd-in-Advanced-Mode を「Y」（Yes）に設定します。詳細については、「*プロファイルの設定*」を参照してください。

- 実行する機能用にコマンドが生成されて画面に表示されるまでは、ワークプランに使用する機能を実行します。

2 PF5 キー（保存）を選択します。

［**Save Workplan**］ウィンドウが表示されます。

3 新しいワークプランの名前と説明を入力し、ENTER キーを押します。

ワークプランが、PROCEDURE ワークプランとしてワークプランライブラリに保存されます。現在の機能のために生成されたコマンドが含まれます。

LIST ワークプランの作成

タイプ LIST のワークプランの作成の詳細については、「*オブジェクトリスト - LIST ワークプラン*」セクションを参照してください。

ワークプランの保存

手順 14.3. ワークプランを保存するには

1 ワークプランの編集が完了したら、コマンド行に次のコマンドを入力します。

```
SAVE
```

または:

PF5 キー（保存）を選択します。

［**Save Workplan**］ウィンドウが表示されます。

2 ワークプランの名前と説明を入力または変更し、ENTER キーを押します。

ワークプランが、指定した名前でワークプランライブラリに保存されます。

ワークプランライブラリの変更

この機能は、ワークプランライブラリを変更するときに使用します。すべてのワークプランはワークプランライブラリに保存する必要があります。

注意: また、オブジェクトハンドラプロファイルで `Workplan-Library` パラメータを指定して、ワークプランのデフォルトライブラリを設定することもできます（「*プロファイルの設定*」を参照）。

［**Change Workplan Library**］画面には、次のフィールドが表示されます。

フィールド	説明
Library	ワークプランライブラリの名前。デフォルトのライブラリは `WORKPLAN` です。
Select library	使用可能なすべてのワークプランライブラリのリストを表示します。「*ライブラリの選択*」も参照してください。 対応する PF キー：PF4 キー（SeLib）
DBID/FNR	ワークプランライブラリが保存されているデータベース ID（DBID）とファイル番号（FNR）を指定します。値が指定されていない場合、現在の FUSER または FNAT システムファイルが使用されます。
Store values in profile	ワークプランライブラリに対して指定された値をオブジェクトハンドラプロファイルに保存するかどうかを決定します。

フィールド	説明	
	N	指定した値は保存しないでください。 これがデフォルトです。
	U	指定した値はユーザー固有のプロファイル設定に保存してください。
	G	指定した値は一般的なプロファイル設定に保存してください。
	「*プロファイルの設定*」も参照してください。	

15 システムファイルの選択

アンロード機能に使用するシステムファイルをリストから選択することができます。

アンロード機能を上級ユーザーモードで実行している場合、オブジェクト指定画面からこの選択リストを作成できます。

以下の手順は、Natural ライブラリオブジェクトをアンロードする場合の機能の使用例です。

手順 15.1. システムファイルをリストから選択するには

1 ［**Unload Natural Library Objects**］画面で［**Select DBID/FNR**］を選択するか PF5 キー（DBID）を押します。

   ［**Select System File**］ウィンドウが表示され、現在の Natural 環境で使用可能なシステムファイルのディレクトリパス名（［**Path**］）、データベース ID（［**DBID**］）、およびファイル番号（［**FNR**］）がリスト表示されます。

2 目的のシステムファイルの横にある［**Sel**］列に任意の単一文字を入力して、機能処理に使用するシステムファイルを選択します。

   FNAT/FUSER を選択すると（デフォルトで選択される）、現在の FNAT および FUSER システムファイルに含まれているすべてのライブラリが使用されます。

   選択したシステムファイルのデータベース ID とファイル番号が、［**Unload Natural Library Objects**］画面の［**DBID/FNR**］フィールドに入力されます。デフォルトのシステムファイルを選択した場合、これらのフィールドは空のままになります。

16 ライブラリの選択

アンロード機能に使用するライブラリをリストから選択することができます。

上級ユーザーモードで実行した場合のアンロード機能のオブジェクト指定画面に表示される［**Select library**］（または［**Select**］）機能には、選択リストが表示されます。

以下の手順は、Natural ライブラリオブジェクトをアンロードする場合の 1 つまたは複数のライブラリの選択例です。

手順 16.1. 1 つのライブラリをリストから選択するには

1 ［**Unload Natural Library Objects**］画面で［**Select library**］を選択するか PF4 キー（SeLib）を押します。

   ［**Select Library**］ウィンドウが開いて、すべてのライブラリ、データベース ID（**DBID**）、およびライブラリが保存されているシステムファイルのファイル番号（**FNR**）がリスト表示されます。

2 目的のライブラリの横にある［**Cmd**］列に、任意の単一文字を入力します。

3 ［ENTER］を選択します。

   ［**Unload Natural Library Objects**］画面の［**Library**］フィールドおよび［**DBID/FNR**］フィールドが、指定の名前および番号でそれぞれ埋められます。［**DBID/FNR**］フィールドに値が入力されていない場合、または「0」が入力されている場合、現在の FUSER または FNAT システムファイルが選択されます。Natural コマンドプロセッサソースに対しては、LFILE 190 の現在の設定が使用されます。

手順 16.2. 複数のライブラリをリストおよび選択するには

1 ［**Unload Natural Library Objects**］画面で［**Select library**］を選択するか PF4 キー（SeLib）を押します。

［**Select Library**］ウィンドウが開いて、すべてのライブラリ、データベースID（**DBID**）、およびライブラリが保存されているシステムファイルのファイル番号（**FNR**）がリスト表示されます。

2 ［**Library**］フィールドに単一の名前または名前の範囲を入力して、選択するライブラリにフィルタをかけます。1つのライブラリ名を入力すると、このライブラリでリストが開始します。有効な名前の範囲については、「*名前と日時の指定*」セクションの「*名前*」を参照してください。

または:

［**DBID**］および［**FNR**］フィールドに、選択するライブラリが含まれているシステムファイルのデータベース ID とファイル番号をそれぞれ入力します。値が入力されていない場合、または「0」が入力されている場合、現在の FUSER または FNAT システムファイルが使用されます。Natural コマンドプロセッサソースに対しては、LFILE 190 の現在の設定が使用されます。

［**Select Library**］ウィンドウが［**Exceptions**］画面から表示された場合、［**DBID**］および［**FNR**］は読み取り専用フィールドになります。

3 ［ENTER］を選択します。

［**Select Library**］ウィンドウに、指定した範囲のすべてのライブラリがリスト表示されます。

4 PF4 キー（Se Rng）を押します。

［**Unload Natural Library Objects**］画面の［**Library**］フィールドおよび［**DBID/FNR**］フィールドが、指定の名前（または範囲）および番号でそれぞれ埋められます。
［**DBID/FNR**］フィールドに値が入力されていない場合、または「0」が入力されている場合、現在の FUSER または FNAT システムファイルが選択されます。

注意: Natural コマンドプロセッサソース（Adabas ファイルに保存される）をアンロードする場合は、DBID および FNR で指定された Adabas ファイルでライブラリが検索されます。DBID および FNR に値が指定されていない場合、または 0 が指定されている場合、LFILE 190 の現在の設定が使用されます。

17 システムエラーメッセージの選択

■ 列およびコマンド 112

アンロードする Natural システムエラーメッセージをリストから選択することができます。

アンロード機能を上級ユーザーモードで実行している場合、オブジェクト指定画面からこの選択リストを作成できます。

手順 17.1. Natural システムエラーメッセージを選択するには

■ ［**Unload Natural System Error Messages**］画面で、必要に応じて［**Error number from/to**］フィールドのメッセージ番号を変更し（デフォルトは数字の全範囲）、［**Select system error messages**］を選択します。

指定のシステムファイルに含まれているすべてのシステムエラーメッセージのリストを示す［**List System Error Messages**］画面が表示されます。

この画面については、次のセクションで説明します。

注意: ユーザー定義エラーメッセージの選択機能については、「*オブジェクトの選択*」セクションで説明します。

列およびコマンド

次の表では、［**List System Error Messages**］画面に表示される列およびコマンドについて説明します。

各列見出しの下にある入力フィールドを使用して、特定のシステムエラーメッセージからリストを開始したり、メッセージにフィルタをかけたりすることができます。有効な入力値については、以下の表を参照してください。

列	PF キー	説明	
Cmd		目的のシステムエラーメッセージの横にある入力フィールドに、次のいずれかの行コマンドを入力できます。	
		L または LI	メッセージの短文テキストおよび長文テキストをリスト表示します。
		S または SE UL または U	以降のアンロード処理のためにメッセージを選択します。 注意： これらのコマンドは、以降の処理のために選択されたメッセージをマークするだけです。アンロード機能を実行するには、PF2 キー（Unloa）を押す必要があります（下記参照）。

列	PF キー	説明
		DE　メッセージを削除します。 DL　メッセージの長文テキストのみを削除します。
Number		システムエラーメッセージの番号。 単一の番号または番号の範囲を入力できます。有効な範囲は次のとおりです。 *value** *value* で開始する番号を持つすべてのメッセージ。 *value*> *value* 以上の番号を持つすべてのメッセージ。 例：10> *value*< *value* 以下の番号を持つすべてのメッセージ。 例：100<
S/L		システムエラーメッセージのテキストの種類： S　短文テキスト。 L　長文テキスト。 A　短文テキストと長文テキストの両方または片方。
Language		システムエラーメッセージの言語コード。 選択するエラーメッセージに対して、8 個までの有効な言語コード（1 は英語など）を入力できます。 アスタリスク（*）にすると、すべての言語コードが選択されます。
Error Message Text		システムエラーメッセージの短文テキスト。
	PF2	処理を行うため、選択されたシステムエラーメッセージのアンロードを開始します。 別の方法としては、コマンド行に次のダイレクトコマンドのいずれかを入力します。 UNLOAD または UNLD
	PF11	PF2 キーを使用した以降のアンロード処理のため、リスト表示されたすべてのシステムエラーメッセージをマークします。 別の方法としては、コマンド行に次のダイレクトコマンドのいずれかを入力します。 SELECT ALL または SEL ALL

18 オブジェクトの選択

■ リスト画面の列およびコマンド 116

アンロードするオブジェクトをリストから選択することができます。この選択リストは、オブジェクトのソースのリスト表示やオブジェクトの削除などの他の目的にも使用できます。

上級ユーザーモードで実行した場合のアンロード機能のオブジェクト指定画面に表示される［**Select objects**］（または［**Select**］）機能には、選択リストが表示されます。

選択リストは、次のセクションで説明する［**List**］画面に表示されます。

注意: Natural システムエラーメッセージの選択機能については、「*システムエラーメッセージの選択*」セクションで説明します。

リスト画面の列およびコマンド

次の表では、［**List**］画面に表示される列およびコマンドについて説明します。

［**List**］画面での列の表示は、［**Select Unload Type**］メニューで選択したオブジェクトのタイプによって異なります。処理されたオブジェクトのタイプは、［**List Library Objects**］や［**List Command Processors**］などのように画面タイトルに示されます。

各列見出しの下にある入力フィールドを使用して、特定のオブジェクトからリストを開始したり、オブジェクトにフィルタをかけたりすることができます。有効な入力値については、以下の表を参照してください。

列	PF キー	説明	
Cmd		目的のオブジェクトの横にある入力フィールドに、次のいずれかの行コマンドを入力できます。	
		L または LI	オブジェクトのソースコードをリスト表示します（Natural コマンドプロセッサソースには適用されません）。 ユーザー定義エラーメッセージの場合：エラーメッセージの短文テキストおよび長文テキストがリスト表示されます。
		S または SE UL または U	以降のアンロード処理のためにオブジェクトを選択します。 注意：これらのコマンドは、以降の処理のために選択されたオブジェクトをマークするだけです。アンロード機能を実行するには、PF2 キー（Unloa）を押す必要があります（下記参照）。
		DE	オブジェクトを削除します。

列	PF キー	説明
Name		オブジェクト名です。 単一の名前または名前の範囲を入力できます。「*名前と日時の指定*」の「*名前*」を参照してください。 ユーザー定義エラーメッセージについては、メッセージ番号および言語コードが表示されます。例えば、次のようになります。`10 (Lang =1)` は、言語 1（英語）のメッセージ番号 10 を表します。
Type		`Program` など、Natural ライブラリオブジェクトのタイプです。 有効な入力値は、1 つまたは複数のオブジェクトタイプコード（プログラムには `P` など）です。コードのリストについては、*`select-clause`*セクションの `NATTYPE` を参照してください。
S/C		Natural ライブラリオブジェクトのタイプ。デフォルトでは、すべてのソース（`S`）オブジェクトまたはカタログされた（`C`）オブジェクトあるいはその両方が画面に表示されます。 有効な入力値は、次の 1 つ以上のコードです。 `S` ソースオブジェクトのみ。 `C` カタログ化オブジェクトのみ。 `S/C` ソースオブジェクトとカタログ化オブジェクトの両方（両方ともある場合）。 `W` STOW を実行された全オブジェクト：同じ日時のソースオブジェクトとカタログ化オブジェクト `*` ソースオブジェクトとカタログ化オブジェクトの両方または片方のすべて。 ユーザー定義エラーメッセージについては、この列にエラーメッセージの短文テキストが表示されます。
M		Natural ライブラリオブジェクトのプログラミングモード。デフォルトでは任意のモードが表示されます。 有効な入力値は、次の 1 つ以上のコードです。 `S` ストラクチャードモードのみ。 `R` レポーティングモードのみ。 `*` 任意のモード。ストラクチャードモードおよびレポーティングモードのいずれかあるいはその両方。 ユーザー定義エラーメッセージについては、この列にエラーメッセージの短文テキストが表示されます。

列	PF キー	説明
Version		Naturalライブラリオブジェクトの保存またはカタログ化あるいはその両方を行ったNaturalのバージョン（範囲指定はできません）。 ユーザー定義エラーメッセージについては、この列にエラーメッセージの短文テキストが表示されます。
User ID		Natural ライブラリオブジェクトを保存またはカタログしたユーザーの ID。 単一のユーザー ID またはユーザー ID の範囲を入力できます。「*名前*」を参照してください。 ユーザー定義エラーメッセージについては、この列にエラーメッセージの短文テキストが表示されます。
Date		Natural ライブラリオブジェクトが保存またはカタログされた日付。単一の日付または日付の範囲を入力できます。「*名前と日時の指定*」の「*日付*」を参照してください。 ユーザー定義エラーメッセージについては、この列にエラーメッセージの短文テキストが表示されます。
Time		Natural ライブラリオブジェクトが保存またはカタログされた時刻。単一の時刻または時刻の範囲を入力できます。「*名前と日時の指定*」の「*時刻*」を参照してください。 ユーザー定義エラーメッセージについては、この列にエラーメッセージの短文テキストが表示されます。

	PF2	処理を行うため、選択されたオブジェクトのアンロードを開始します。
		
		別の方法としては、コマンド行に次のダイレクトコマンドのいずれかを入力します。
		
		`UNLOAD`
		または
		`UNLD`

	PF11	**PF2** キーによる後続のアンロード処理のために、リスト表示されたすべてのオブジェクトをマークします。
		
		別の方法としては、コマンド行に次のダイレクトコマンドのいずれかを入力します。
		
		`SELECT ALL`
		または
		`SEL ALL`

19 オブジェクト指定

オブジェクトハンドラのメインメニューにより［**Select Unload/Load/Scan Type**］画面が表示されます。この画面で、処理するオブジェクトのタイプを選択するか、タイプ SELECTION または LIST のワークプランを指定できます。

選択したオブジェクトのタイプごとに、個々のオブジェクト指定画面が表示されます。これらの画面を使用して、処理するオブジェクトの選択条件を指定します。

注意: 時間のかからない代替手段として、上級ユーザーはコンパクトモードを使用できます。「*コンパクトモード*」を参照してください。

このセクションでは、各オブジェクト指定画面に表示されるオプションについて説明します。このセクションで説明するフィールドまたはファンクションキー（PF キー）が特定の機能や上級ユーザーモードでしか表示されない場合は、「上級ユーザーモードのアンロード機能にのみ適用されます。」などの適切な注釈によって示されます。

ワークファイルのすべてのオブジェクト

Natural ライブラリオブジェクト

Natural システムエラーメッセージ

Natural コマンドプロセッサソース

Natural 関連オブジェクト

外部オブジェクト

FDT

SELECTION または **LIST** ワークプランの使用

20 オブジェクト指定 - ワークファイルのすべてのオブジェクト

ロード機能またはスキャン機能にのみ適用されます。

［**Load/Scan All Objects on the Work File**］オプションは、ワークファイルで処理可能なすべてのオブジェクトを選択するために使用します。上級ユーザーモードでは、［**Load/Scan All Objects**］画面から［**Settings**］画面を表示して、この画面でオプションの指定およびパラメータの設定を行うことができます。「*設定*」セクションを参照してください。キーワードと有効な値については、「*ダイレクトコマンド*」セクションの `select-clause` を参照してください。

21 オブジェクト指定 - Natural ライブラリオブジェクト

- Natural ライブラリオブジェクト 124
- Natural ライブラリオブジェクトの詳細 125
- Natural ライブラリオブジェクトのプロパティ 128
- Natural ライブラリオブジェクトの例外 129
- Natural ライブラリオブジェクトの例外のプロパティ 130

このセクションでは、Naturalライブラリオブジェクトを処理するために使用する、オブジェクト指定画面に表示されるオプションについて説明します。Natural ライブラリオブジェクトとは、プログラミングオブジェクト（Natural DDM を含む）、ユーザー定義エラーメッセージ、および共有リソースのことです。

キーワードと有効な入力値の詳細については、「*ダイレクトコマンド*」セクションの `select-clause` に関する記載も参照してください。

Natural ライブラリオブジェクト

［**Unload/Load/Scan Natural Library Objects**］画面には、次のフィールドおよび PF キーが表示されます。

フィールド	PF キー	説明
DBID/FNR		アンロード機能にのみ適用されます。 Natural ライブラリが保存されているシステムファイルのデータベース ID（DBID）とファイル番号（FNR）。値が指定されていない場合、または 0 が指定されている場合、現在の FUSER または FNAT システムファイルが使用されます。
Select DBID/FNR	PF5 （上級ユーザーモード専用）	アンロード機能にのみ適用されます。 使用可能なシステムファイルのリストを表示します。
Library		ライブラリの名前または名前の範囲。「*名前と日時の指定*」の「*名前*」を参照してください。
Select library	PF4	使用可能なすべてのライブラリの選択リストが表示されます。「*ライブラリの選択*」も参照してください。
Object name		Natural プログラミングオブジェクトまたは共有リソースの名前、あるいは名前の範囲。「*名前*」を参照してください。 画面［**Natural Library Objects, Details**］で、フィールド［**Natural programming objects**］（デフォルト設定）または［**Shared resources**］あるいはその両方が選択されている場合にのみ評価されます。「***Natural** ライブラリオブジェクトの詳細*」も参照してください。
Select objects		上級ユーザーモードのアンロード機能にのみ適用されます。 ライブラリの範囲が指定されていない場合、使用可能なすべての Natural オブジェクトの選択リストが表示されます（「*オブジェクトの選択*」も参照）。

フィールド	PF キー	説明
Error number from/to		最初と最後のメッセージ番号で定められたユーザー定義エラーメッセージの有効な範囲（1～9999）。 ［**Natural Library Objects, Details**］画面で［**Error messages**］フィールド（デフォルト設定）が選択されている場合にのみ評価されます（「*Natural ライブラリオブジェクトの詳細*」も参照）。
Details	PF6	［**Natural Library Objects, Details**］画面が表示され、この画面でオブジェクトの詳細指定を入力できます。「*Natural ライブラリオブジェクトの詳細*」を参照してください。
Settings	PF7	上級ユーザーモードで実行されている機能にのみ適用されます。 ［**Unload/Load/Scan Settings**］画面が表示され、この画面でオプションとパラメータ設定を指定できます。「*設定*」を参照してください。
Work file	PF11 PF1	その機能を使用するワークファイルの名前。 名前が使用可能なスペースを超える場合、PF11 キー（WorkF）を押すと名前をさらに長く入力できます（最大 253 文字）。または、このフィールドでカーソルを表示して PF1 キー（ヘルプ）を押します。 「*ワークファイル*」も参照してください。

Natural ライブラリオブジェクトの詳細

画面［**Unload/Load/Scan Natural Library Objects, Details**］は、Natural ライブラリオブジェクトの選択条件を詳細に指定するために使用します。

キーワードと有効な入力値の詳細については、「*ダイレクトコマンド*」セクションの `select-clause` に関する記載も参照してください。

［**Unload/Load/Scan Natural Library Objects, Details**］画面には、次のフィールドおよび PF キーが表示されます。

フィールド	PF キー	説明
Library		ライブラリの名前または名前の範囲。「*名前と日時の指定*」の「*名前*」を参照してください。 ［**Use Predict set**］オプションを選択した場合、範囲指定はできません。
Select （ライブラリ）	PF4	使用可能なすべてのライブラリの選択リストが表示されます。「*ライブラリの選択*」も参照してください。
DBID/FNR		上記の「*Natural ライブラリオブジェクト*」の「**DBID/FNR**」を参照してください。

フィールド	PF キー	説明
Object Types: **Natural programming objects**		Natural DDM（データ定義モジュール）を含む Natural プログラミングオブジェクト。 FDDM システムファイルが有効になっている場合、「*設定*」セクションの「**Use FDDM file for processing DDMs**」の項も参照してください。
Object Types: **Error messages**		ユーザー定義エラーメッセージ。
Object Types: **Shared resources**		Natural 環境で使用され、Natural ライブラリシステムでメンテナンスされている Natural 以外のあらゆるファイル。 メインフレームプラットフォーム上のリモート環境では、共有リソースは定義されません。
Object name		上記の「*Natural ライブラリオブジェクト*」の「**Object name**」を参照してください。
Use Predict set		Predict がインストールされている場合に、アンロード機能および検索機能にのみ適用されます。 このオプションは、処理するオブジェクトの名前を保有セットから読み取るために使用します。保有セットは、`LIST XREF` コマンドのセット保存オプションで作成されます。 ［**Use Predict set**］オプションを選択すると、次の制限が適用されます。 ■ ［**Object name**］フィールドに、すべてのオブジェクトを示すアスタリスク（*）が含まれている必要があります。これはデフォルト設定です。 ■ ［**Library**］フィールドに、1 つのライブラリの名前が含まれている必要があります。名前の範囲は使用できません。 ■ ［**Set number**］フィールドに、値が入力されている必要があります。 Predict セットの詳細については、『*Predict*』ドキュメントを参照してください。
Set number		適用されるのは、［**Use Predict set**］が選択されている場合に限られます。 使用する保有セットを識別する 1 桁または 2 桁の数字。
Set library		適用されるのは、［**Use Predict set**］が選択されている場合に限られます。 Predict セットに使用するために検索するライブラリの名前。名前を指定しない場合、［**Library**］フィールドに入力したライブラリがデフォルトで使用されます。
Set user		適用されるのは、［**Use Predict set**］が選択されている場合に限られます。 保有セットを作成したユーザーの ID。ID が入力されていない場合、システム変数 `*USER` が指定された ID が使用されます（『システム変数』ドキュメントを参照）。

フィールド	PF キー	説明
Programming Object Options: **S/C-Kind**		Natural プログラミングオブジェクトの種類： `S` ソースオブジェクトのみ。 `C` カタログ化オブジェクトのみ。 `A` または `*` ソースオブジェクトとカタログ化オブジェクトの両方または片方のすべて。これはデフォルト設定です。 `W` STOW を実行された全オブジェクト：同じ日時のソースオブジェクトとカタログ化オブジェクト `B` ソースオブジェクトとカタログ化オブジェクトの両方（両方ともある場合）。 **注意:** `W` と `B` は、アンロード機能でのみ有効です。`W` と `B` をロード機能またはスキャン機能で入力することもできますが、`A` として処理されます。
Programming Object Options: **Natural types**		Natural オブジェクトタイプのコード（プログラムの場合は `P` など）。有効なコードのリストについては、 *`select-clause`* セクションの `NATTYPE` を参照してください。
Select Natural types	PF6	1 つまたは複数の Natural オブジェクトタイプを選択できるウィンドウが表示されます。
Properties	PF7	Natural プログラミングオブジェクトの追加プロパティを指定できる別の画面が表示されます。「***Natural** ライブラリオブジェクトのプロパティ*」を参照してください。
Error Messages: **Error number from/to**		［**Error number from/to**］フィールドに入力したユーザー定義エラーメッセージの範囲（上記の「*Natural ライブラリオブジェクト*」を参照）。
Error Messages: **Language codes**		指定されたエラーメッセージの有効な言語コード（英語の場合はコード `1`）。最大 8 つまで指定可能。 アスタリスク（*）にすると、すべての言語コードが選択されます。
Error Messages: **S/L-Kind**		エラーメッセージのテキストの種類： `S` 短文テキスト。 `L` 長文テキスト。 `A` 短文テキストと長文テキストの両方または片方。これがデフォルトです。 `B` 両方とも存在する場合は短文テキストと長文テキスト（アンロード機能のみ）。

フィールド	PF キー	説明
Exceptions	PF8	Naturalプログラミングオブジェクトの選択の例外を指定できる別の画面が表示されます。「*Natural ライブラリオブジェクトの例外*」を参照してください。

Natural ライブラリオブジェクトのプロパティ

［**Unload/Load/Scan Library Objects, Properties**］画面は、処理を行うために選択した Natural ライブラリオブジェクトのプロパティを指定するために使用します。

キーワードと有効な入力値の詳細については、「*ダイレクトコマンド*」セクションの `select-clause` に関する記載も参照してください。

［**Unload/Load/Scan Library Objects, Properties**］画面には、次のフィールドが表示されます。

フィールド	説明
User ID	Natural プログラミングオブジェクトを保存またはカタログしたユーザーの ID。単一のユーザー ID またはユーザー ID の範囲を指定します。「*名前と日時の指定*」の「*名前*」を参照してください。
Programming mode	Natural プログラミングオブジェクトのプログラミングモード： R：レポーティングモードのみ。 S：ストラクチャードモードのみ。 A：モードチェックは実行されません。これはデフォルト設定です。
Natural version	Natural プログラミングオブジェクトの Natural バージョン。 バージョンの範囲を指定することもできます。「*名前*」を参照してください。
DDM DBID	データ定義モジュール（DDM）のデータベース ID（DBID）。 有効なエントリ：1～65535、または 0（すべての DBID）
DDM FNR	DDM のファイル番号（FNR）： 有効なエントリ：1～65535、または 0（すべての FNR）
Object Date: **Select all objects (no date check)**	日付に関係なく、すべてのオブジェクトを選択します。
Object Date: **Select objects modified between/and**	一定の範囲内にある保存またはカタログした日付または時刻（あるいはその両方）のすべてのオブジェクトを選択します。範囲を指定するには、該当するフィールドで開始日付または開始時刻（あるいはその両方）と終了日付または終了時刻（あるいはその両方）を指定します。

フィールド	説明
	有効な入力値については、「*名前と日時の指定*」の「*日付*」および「*時刻*」を参照してください。日付には、次の特殊な入力値を使用できます。TODAY、YESTERDAY、MONTH、および YEAR
Object Date: **Select objects modified on**	正確な日付または時刻（あるいはその両方）を入力することによってこのフィールドで指定された日時に一致する、保存またはカタログした日付または時刻（あるいはその両方）のすべてのオブジェクトを選択します。 有効な入力値については、「*日付*」および「*時刻*」を参照してください。日付には、次の特殊な入力値を使用できます。TODAY および YESTERDAY
Object Size: **Select all objects (no size check)**	サイズに関係なく、すべてのオブジェクトを選択します。
Object Size: **Select objects with size between/and**	一定の範囲内にあるサイズのすべてのオブジェクトを選択します。範囲を指定するには、該当するフィールドで下限のサイズと上限のサイズを指定します。
Object Size: **Select objects with size**	このフィールドで指定されたサイズに一致するすべてのオブジェクトを選択します。

Natural ライブラリオブジェクトの例外

［**Unload/Load/Scan Library Objects, Exceptions**］画面は、Natural ライブラリオブジェクトの選択の例外を指定するために使用します。

「***Natural** ライブラリオブジェクト*」、「***Natural** ライブラリオブジェクトの詳細*」、および「***Natural** ライブラリオブジェクトのプロパティ*」で指定した選択条件に一致するすべてのオブジェクトは、［**Unload/Load/Scan Library Objects, Exceptions**］画面で行った指定に対してチェックされます。例外として定義されたすべての指定項目に一致するオブジェクトは、処理から除外されます。

キーワードと有効な入力値の詳細については、「*ダイレクトコマンド*」セクションの *select-clause* に関する記載も参照してください。

［**Unload/Load/Scan Library Objects, Exceptions**］画面は、［**Unload/Load/Scan Natural Library Objects, Details**］画面と基本的に同じです。下の表に示すフィールド、コマンド、および代替 PF キーの該当するセクションの説明を参照してください。［**Add/change properties for selection**］フィールドは、Natural プログラミングオブジェクトの例外に関して追加プロパティを指定するために使用します。「***Natural** ライブラリオブジェクトの例外のプロパティ*」を参照してください。

フィールド	PF キー
Library	PF4
Select （ライブラリ）	
Object Types: **Natural programming objects** **Error messages** **Shared resources**	
Object name	
S/C-Kind	
Natural types	
Select Natural types	PF6
Properties	PF7
Error number	
S/L-Kind	
Languages	

Natural ライブラリオブジェクトの例外のプロパティ

［**Unload/Load/Scan Library Objects, Exceptions**］画面は、処理を行うために選択した Natural ライブラリオブジェクトのプロパティの例外を指定するために使用します。

画面には、次のフィールドが表示されます。

フィールド	説明
User ID	「*Natural ライブラリオブジェクトのプロパティ*」の「**User ID**」を参照してください。
Programming mode	「*Natural ライブラリオブジェクトのプロパティ*」の「**Programming mode**」を参照してください。
Natural version	「*Natural ライブラリオブジェクトのプロパティ*」の「**Natural version**」を参照してください。
DDM DBID	「*Natural ライブラリオブジェクトのプロパティ*」の「**DDM DBID**」を参照してください。
DDM FNR	「*Natural ライブラリオブジェクトのプロパティ*」の「**DDM FNR**」を参照してください。
Object Date: **Ignore object date**	日付チェックは実行されません。日付に関係なく、オブジェクトは処理されます。

フィールド	説明
Object Date: **Exclude objects modified between/and**	一定の範囲内にある保存またはカタログした日付または時刻（あるいは両方）のすべてのオブジェクトを処理から除外します。範囲を指定するには、該当するフィールドで開始日付または開始時刻（あるいは両方）と終了日付または終了時刻（あるいは両方）を指定します。 有効な入力値については、「*名前と日時の指定*」の「*日付*」および「*時刻*」を参照してください。日付には、次の特殊な入力値を使用できます。TODAY、YESTERDAY、MONTH、および YEAR
Object Date: **Exclude objects modified on**	正確な日付または時刻（あるいはその両方）を入力することによってこのフィールドで指定された日時に一致する、保存またはカタログした日付または時刻（あるいはその両方）のすべてのオブジェクトを処理から除外します。 有効な入力値については、「*日付*」および「*時刻*」を参照してください。日付には、次の特殊な入力値を使用できます。TODAY および YESTERDAY
Object Size: **Ignore object size**	サイズチェックは実行されません。サイズに関係なく、オブジェクトは処理されます。
Object Size: **Exclude objects with size between/and**	一定の範囲内にあるサイズのすべてのオブジェクトを処理から除外します。範囲を指定するには、該当するフィールドで下限のサイズと上限のサイズを指定します。
Object Size: **Exclude objects with size**	このフィールドで指定されたサイズに一致するすべてのオブジェクトを処理から除外します。

22 オブジェクト指定 - Natural システムエラーメッセージ

- Natural システムエラーメッセージ 134
- Natural システムエラーメッセージの詳細 134
- Natural システムエラーメッセージの例外 135

このセクションでは、現在のFNATファイルまたはワークファイルからNaturalシステムエラーメッセージを処理するために使用する、オブジェクト指定画面に表示されるオプションについて説明します。

キーワードと有効な入力値の詳細については、「*ダイレクトコマンド*」セクションの `select-clause` に関する記載も参照してください。

Natural システムエラーメッセージ

［**Unload/Load/Scan Natural System Error Messages**］画面には、次のフィールドおよび PF キーが表示されます。

フィールド	PF キー	説明
Error number from/to		最初と最後のメッセージ番号で定められた Natural システムエラーメッセージの範囲。使用可能なすべてのシステムエラーメッセージを表示するには、［ **Select system error messages**］を選択します。
Details	PF6	オブジェクト指定をより詳細に入力できる［**Unload/Load/Scan Natural Library Objects, Details**］が開きます。「***Natural*** *システムエラーメッセージの詳細*」を参照してください。
Settings	PF7	［**Unload/Load/Scan Settings**］画面が表示され、この画面でオプションとパラメータ設定を指定できます。「*設定*」を参照してください。
Work file	PF11 PF1	この機能に使用するワークファイルの名前。「*Natural ライブラリオブジェクト*」セクションの「**Work file**」も参照してください。

Natural システムエラーメッセージの詳細

画面［**Unload/Load/Scan System Error Messages, Details**］は、Natural システムエラーメッセージの選択条件を詳細に指定するために使用します。

キーワードと有効な入力値の詳細については、「*ダイレクトコマンド*」セクションの `select-clause` に関する記載も参照してください。

［**Unload/Load/Scan System Error Messages, Details**］画面には、次のフィールドおよび PF キーが表示されます。

フィールド	PF キー	説明
Error number from/to		上記の「*Natural システムエラーメッセージ*」の「**Error number**」を参照してください。
Language codes		「*Natural ライブラリオブジェクトの詳細*」の「**Language codes**」を参照してください。
S/L-Kind		「*Natural ライブラリオブジェクトの詳細*」の「**S/L-Kind**」を参照してください。
Exceptions	PF8	Natural システムエラーメッセージの選択の例外を指定できる別の画面が表示されます。「***Natural** システムエラーメッセージの例外*」を参照してください。

Natural システムエラーメッセージの例外

［**Unload/Load/Scan System Error Messages, Exceptions**］画面は、Natural システムエラーメッセージの選択の例外を指定するために使用します。

「***Natural** システムエラーメッセージ*」および「***Natural** システムエラーメッセージの詳細*」で指定した選択条件に一致するすべての Natural システムエラーメッセージは、［**Unload/Load/Scan System Error Messages, Exceptions**］画面で行った指定に対してチェックされます。例外として定義されたすべての指定項目に一致するエラーメッセージは、処理から除外されます。

例外画面に表示されるフィールドの説明については、上記の「***Natural** システムエラーメッセージの詳細*」を参照してください。

キーワードと有効な入力値の詳細については、「*ダイレクトコマンド*」セクションの `select-clause` に関する記載も参照してください。

23 オブジェクト指定 - Natural コマンドプロセッサ

- Natural コマンドプロセッサ 138
- Natural コマンドプロセッサソースの例外 139

このセクションでは、Natural コマンドプロセッサソース（Adabas ファイルに保存される）を処理するために使用する、オブジェクト指定画面に表示されるオプションについて説明します。

キーワードと有効な入力値の詳細については、「*ダイレクトコマンド*」セクションの `select-clause` に関する記載も参照してください。

Natural コマンドプロセッサ

［**Unload/Load/Scan Natural Command Processors**］画面には、次のフィールドおよび PF キーが表示されます。

フィールド	PF キー	説明
Library		Natural コマンドプロセッサライブラリの名前または名前の範囲。「*名前と日時の指定*」の「*名前*」を参照してください。
Select library	PF4	使用可能な Natural コマンドプロセッサライブラリの選択リストを表示します。「*ライブラリの選択*」も参照してください。
DBID/FNR		アンロード機能にのみ適用されます。 Natural コマンドプロセッサソースが保存されている Adabas（FNR）ファイルのデータベース ID（DBID）とファイル番号。 値が指定されていない場合、LFILE 190 の現在の設定が使用されます。詳細については、『ユーティリティ』ドキュメントの「***SYSNCP** ユーティリティ*」を参照してください。
Password/Cipher		アンロード機能にのみ適用されます。 Natural コマンドプロセッサソースが保存されている Adabas ファイルのパスワードとサイファコード。
Object name		Natural コマンドプロセッサソースの名前または名前の範囲。「*名前*」を参照してください。
Select objects		アンロード機能にのみ適用されます。 ライブラリの範囲が指定されていない場合にこのフィールドを選択すると、使用可能な Natural コマンドプロセッサソースの選択リストが表示されます（「*オブジェクトの選択*」も参照）。
Exceptions	PF8	Natural コマンドプロセッサソースの選択の例外を指定できる別の画面が表示されます。「***Natural** コマンドプロセッサソースの例外*」を参照してください。
Settings	PF7	［**Unload/Load/Scan Settings**］画面が表示され、この画面でオプションとパラメータ設定を指定できます。「*設定*」を参照してください。
Work file	PF11 PF1	この機能に使用するワークファイルの名前。「*Natural ライブラリオブジェクト*」セクションの「**Work file**」も参照してください。

Natural コマンドプロセッサソースの例外

［**Unload/Load/Scan Natural Command Processors, Exceptions**］画面は、Natural コマンドプロセッサソースの選択の例外を指定するために使用します。

「*Natural コマンドプロセッサソース*」で指定した選択条件に一致するすべてのオブジェクトは、［**Unload/Load/Scan Natural Command Processors, Exceptions**］画面で行った指定に対してチェックされます。例外として定義されたすべての指定項目に一致する Natural コマンドプロセッサソースは、処理から除外されます。

例外ウィンドウに表示されるフィールドの説明については、上記の「*Natural コマンドプロセッサソース*」を参照してください。

キーワードと有効な入力値の詳細については、「*ダイレクトコマンド*」セクションの `select-clause` に関する記載も参照してください。

24 オブジェクト指定 - Natural 関連オブジェクト

- Natural 関連オブジェクト 142
- Natural 関連オブジェクトの詳細 143
- Natural 関連オブジェクトの例外 144

適用されるのは、内部フォーマットでデータを処理する場合、言い換えると、［**Transfer format**］が*選択されていない*場合に限られます。「*ワークファイル*」セクションの「*ワークファイルフォーマット*」も参照してください。

このセクションでは、Natural関連オブジェクトを処理するために使用する、オブジェクト指定画面に表示されるオプションについて説明します。Natural 関連オブジェクトとは、Natural 環境に保存されるが、Natural ライブラリおよび Adabas ファイルには保存されないオブジェクトのことです。例として、Natural パス PARM_PATH に存在する NATPARM パラメータファイルがあります。

キーワードと有効な入力値の詳細については、「*ダイレクトコマンド*」セクションの `select-clause` に関する記載も参照してください。

Natural 関連オブジェクト

［**Unload/Load/Scan Natural-Related Objects**］画面には、次のフィールドおよび PF キーが表示されます。

フィールド	PF キー	説明
Natural path		Natural 関連オブジェクトが保存されているパスの名前。 ロードとスキャン： パス名、またはすべてのパスを選択する場合はアスタリスク（*）。 有効な入力値は次のとおりです。 `NATROOT`, `NATBIN`, `NATERR`, `NATSAG`, `PARM_PATH`, `PROFILE_PATH`, `TEXT_PATH`, `TMP_PATH`.
Select Natural path	PF4	使用可能な Natural パスの選択リストを表示します。
Object name	PF5 PF1	Natural 関連オブジェクトの名前。 ロード／スキャンの対象： 単一の名前または名前の範囲。「*名前と日時の指定*」の「*名前*」を参照してください。 名前が使用可能なスペースを超える場合、PF5 キー（Objct）を押すと名前をさらに長く入力できます（最大 253 文字）。または、このフィールドでカーソルを表示して PF1 キー（ヘルプ）を押します。
Details	PF6	オブジェクト指定の詳細を入力できる［**Natural-related Objects, Details**］画面が表示されます。「***Natural** 関連オブジェクトの詳細*」を参照してください。
Settings	PF7	［**Unload/Load/Scan Settings**］画面が表示され、この画面でオプションとパラメータ設定を指定できます。「*設定*」を参照してください。

フィールド	PF キー	説明
Work file	PF11 PF1	この機能に使用するワークファイルの名前。「*オブジェクト指定*」の「*Natural ライブラリオブジェクト*」にある「**Work file**」の項も参照してください。

Natural 関連オブジェクトの詳細

画面［**Unload/Load/Scan Natural-related Objects, Details**］は、Natural 関連オブジェクトの選択条件を詳細に指定するために使用します。

キーワードと有効な入力値の詳細については、「*ダイレクトコマンド*」セクションの `select-clause` に関する記載も参照してください。

［**Unload/Load/Scan Natural-related Objects, Details**］画面には、次のフィールドおよび PF キーが表示されます。

フィールド	PF キー	説明
Natural path		上記の「*Natural 関連オブジェクト*」の「**Natural path**」を参照してください。
Select Natural path	PF4	使用可能な Natural パスの選択リストを表示します。
Object name	PF5	上記の「*Natural 関連オブジェクト*」の「**Object name**」を参照してください。
Object Date: **Select all objects (no date check)**		日付に関係なく、すべてのオブジェクトを選択します。
Object Date: **Select objects modified between/and**		「*Natural ライブラリオブジェクトのプロパティ*」の「**Object Date**」を参照してください。
Object Date: **Select objects modified on**		「*Natural ライブラリオブジェクトのプロパティ*」の「**Object Date**」を参照してください。
Object Size: **Select all objects (no size check)**		サイズに関係なく、すべてのオブジェクトを選択します。
Object Size: **Select objects with size between/and**		一定の範囲内にあるサイズのすべてのオブジェクトを選択します。範囲を指定するには、該当するフィールドで下限のサイズと上限のサイズを指定します。
Object Size:		このフィールドで指定されたサイズに一致するすべてのオブジェクトを選択します。

フィールド	PF キー	説明
Select objects with size		
Exceptions	PF8	Natural 関連オブジェクトの選択の例外を指定できる別の画面が表示されます。「*Natural 関連オブジェクトの例外*」を参照してください。

Natural 関連オブジェクトの例外

［**Unload/Load/Scan Natural-related Objects, Exceptions**］画面は、Natural 関連オブジェクトの選択の例外を指定するために使用します。

「*Natural 関連オブジェクト*」および「*Natural 関連オブジェクトの詳細*」で指定した選択条件に一致するすべての Natural 関連オブジェクトは、［**Unload/Load/Scan Natural-related Objects, Exceptions**］画面で行った指定に対してチェックされます。例外として定義されたすべての指定項目に一致するオブジェクトは、処理から除外されます。

キーワードと有効な入力値の詳細については、「*ダイレクトコマンド*」セクションの `select-clause` に関する記載も参照してください。

［**Unload/Load/Scan Natural-related Objects, Exceptions**］画面には、次のフィールドおよび PF キーが表示されます。

フィールド	PF キー	説明
Natural path	PF4	上記の「*Natural 関連オブジェクト*」の「**Natural path**」を参照してください。
Select Natural path		使用可能な Natural パスの選択リストを表示します。
Object name	PF5	上記の「*Natural 関連オブジェクト*」の「**Object name**」を参照してください。
Object Date: **Ignore object date**		日付チェックは実行されません。日付に関係なく、オブジェクトは処理されます。
Object Date: **Exclude objects modified between/and**		「*Natural ライブラリオブジェクトの例外のプロパティ*」の「**Object Date**」を参照してください。
Object Date: **Exclude objects modified on**		「*Natural ライブラリオブジェクトの例外のプロパティ*」の「**Object Date**」を参照してください。
Object Size: **Ignore object size**		サイズチェックは実行されません。サイズに関係なく、オブジェクトは処理されます。

フィールド	PF キー	説明
Object Size: **Exclude objects with size between/and**		一定の範囲内にあるサイズのすべてのオブジェクトを処理から除外します。範囲を指定するには、該当するフィールドで下限のサイズと上限のサイズを指定します。
Object Size: **Exclude objects with size**		このフィールドで指定されたサイズに一致するすべてのオブジェクトを処理から除外します。

25 オブジェクト指定 - 外部オブジェクト

- 外部オブジェクト 148
- 外部オブジェクトの詳細 149
- 外部オブジェクトの例外 150

適用されるのは、内部フォーマットでデータを処理する場合、言い換えると、［**Transfer format**］が*選択されていない*場合に限られます。「*ワークファイル*」セクションの「*ワークファイルフォーマット*」も参照してください。

このセクションでは、外部オブジェクトを処理するために使用する、オブジェクト指定画面に表示されるオプションについて説明します。外部オブジェクトとは、ビットマップなどのように、Natural 環境および Adabas 環境の外に保存されているオブジェクトのことです。

キーワードと有効な入力値の詳細については、「*ダイレクトコマンド*」セクションの `select-clause` に関する記載も参照してください。

外部オブジェクト

［**Unload/Load/Scan External Objects**］画面には、次のフィールドおよび PF キーが表示されます。

フィールド	PF キー	説明
External path	PF4 PF1	外部オブジェクトが保存されているパスの名前。 ロードとスキャン： パス名、またはすべてのパスを選択する場合はアスタリスク（*）。 名前が使用可能なスペースを超える場合、PF4 キー（Path）を押すと名前をさらに長く入力できます（最大 253 文字）。または、このフィールドでカーソルを表示して PF1 キー（ヘルプ）を押します。
Object name	PF5 PF1	外部オブジェクトの名前。 ロードとスキャン： 単一の名前または名前の範囲。「*名前と日時の指定*」の「*名前*」を参照してください。 名前が使用可能なスペースを超える場合、PF5 キー（Objct）を押すと名前をさらに長く入力できます（最大 253 文字）。または、このフィールドでカーソルを表示して PF1 キー（ヘルプ）を押します。
Details	PF6	オブジェクト指定の詳細を入力できる［**Unload/Load/Scan External Objects, Details**］画面が表示されます。「*外部オブジェクトの詳細*」を参照してください。
Settings	PF7	［**Unload/Load/Scan Settings**］画面が表示され、この画面でオプションとパラメータ設定を指定できます。「*設定*」セクションを参照してください。
Work file	PF11 PF1	この機能に使用するワークファイルの名前。「*Natural ライブラリオブジェクト*」の「**Work file**」も参照してください。

外部オブジェクトの詳細

画面［**Unload/Load/Scan External Objects, Details**］は、外部オブジェクトの選択条件を詳細に指定するために使用します。

キーワードと有効な入力値の詳細については、「*ダイレクトコマンド*」セクションの `select-clause` に関する記載も参照してください。

［**Unload/Load/Scan External Objects, Details**］画面には、次のフィールドおよび PF キーが表示されます。

フィールド	PF キー	説明
External path	PF4	上記の「*外部オブジェクト*」の「**External path**」を参照してください。
Object name	PF5	上記の「*外部オブジェクト*」の「**Object name**」を参照してください。
Object Date: **Select all objects (no date check)**		日付に関係なく、すべてのオブジェクトを選択します。
Object Date: **Select objects modified between/and**		「*Natural ライブラリオブジェクトのプロパティ*」の「**Object Date**」を参照してください。
Object Date: **Select objects modified on**		「*Natural ライブラリオブジェクトのプロパティ*」の「**Object Date**」を参照してください。
Object Size: **Select all objects (no size check)**		サイズに関係なく、すべてのオブジェクトを選択します。
Object Size: **Select objects with size between/and**		「*Natural ライブラリオブジェクトのプロパティ*」の「**Object Size**」を参照してください。
Object Size: **Select objects with size**		このフィールドで指定されたサイズに一致するすべてのオブジェクトを選択します。
Exceptions	PF8	外部オブジェクトの選択の例外を指定できる別の画面が表示されます。「*外部オブジェクトの例外*」を参照してください。

外部オブジェクトの例外

［**Unload/Load/Scan External Objects, Exceptions**］画面は、外部オブジェクトの選択の例外を指定するために使用します。

「*外部オブジェクト*」および「*外部オブジェクトの詳細*」で指定した選択条件に一致するすべての外部オブジェクトは、［**Unload/Load/Scan External Objects, Exceptions**］画面で行った指定に対してチェックされます。例外として定義されたすべての指定項目に一致するオブジェクトは、処理から除外されます。

キーワードと有効な入力値の詳細については、「*ダイレクトコマンド*」セクションの `select-clause` に関する記載も参照してください。

［**Unload/Load/Scan External Objects, Exceptions**］画面には、次のフィールドおよび PF キーが表示されます。

フィールド	PF キー	説明
External path	PF4	上記の「*外部オブジェクト*」の「**External path**」を参照してください。
Object name	PF5	上記の「*外部オブジェクト*」の「**Object name**」を参照してください。
Object Date: **Ignore object date**		日付チェックは実行されません。日付に関係なく、オブジェクトは処理されます。
Object Date: **Exclude objects modified between/and**		「*Natural ライブラリオブジェクトの例外のプロパティ*」の「**Object Date**」を参照してください。
Object Date: **Exclude objects modified on**		「*Natural ライブラリオブジェクトの例外のプロパティ*」の「**Object Date**」を参照してください。
Object Size: **Ignore object size**		サイズチェックは実行されません。サイズに関係なく、オブジェクトは処理されます。
Object Size: **Exclude objects with size between/and**		一定の範囲内にあるサイズのすべてのオブジェクトを処理から除外します。範囲を指定するには、該当するフィールドで下限のサイズと上限のサイズを指定します。
Object Size: **Exclude objects with size**		このフィールドで指定されたサイズに一致するすべてのオブジェクトを処理から除外します。

26 オブジェクト指定 - FDT

このセクションでは、Adabas FDT（フィールド定義テーブル）を処理するために使用する、オブジェクト指定画面に表示されるオプションについて説明します。

注意: FDTをロードすると、すべてのFDTデータはワークファイル5に書き込まれます。このワークファイルの内容は、Adabas ユーティリティ ADAFDU の入力として使用できます。

キーワードと有効な入力値の詳細については、「*ダイレクトコマンド*」セクションの `select-clause` に関する記載も参照してください。

［**Unload/Load/Scan FDTs**］画面には、次のフィールドおよび PF キーが表示されます。

フィールド	PF キー	説明
DBID		FDT が保存されているデータベースの ID。 ロードとスキャン： 有効な DBID、またはすべての DBID の場合は 0。
FNR		FDT が保存されているファイルの番号。 ロードとスキャン： 有効な FNR、またはすべての FDT の場合は 0。
Password/Cipher		アンロードおよびロード機能にのみ適用されます。 FDT が保存されている Adabas ファイルの Adabas パスワードとサイファコード。
Settings	PF7	［**Unload/Load/Scan Settings**］画面が表示され、この画面でオプションとパラメータ設定を指定できます。「*設定*」を参照してください。
Work file	PF11 PF1	この機能に使用するワークファイルの名前。「*Natural ライブラリオブジェクト*」の「**Work file**」も参照してください。

27 SELECTION または LIST ワークプランの使用

このオプションは、タイプ SELECTION または LIST のワークプランを指定するために使用します。これらのワークプランが、処理するオブジェクトの選択条件を指定します。「*ワークプラン*」セクションも参照してください。

［**Unload/Load/Scan Selection or List**］画面には、次のフィールドおよび PF キーが表示されます。

フィールド	PF キー	説明
Name	PF4	処理するワークプランの名前。
List Workplan		［**Name**］フィールドに指定されたワークプランの内容を表示します。
Select Workplan	PF5	使用可能なすべてのワークプランのリストを表示します。「*管理*」の「*ワークプランライブラリ内の利用可能なワークプランのリストの作成*」も参照してください。
Settings	PF7	［**Unload/Load/Scan Settings**］画面が表示され、この画面でオプションとパラメータ設定を指定できます。「*設定*」を参照してください。
Work file	PF11 PF1	この機能に使用するワークファイルの名前。「*Natural ライブラリオブジェクト*」の「**Work file**」も参照してください。

28 設定

- 設定画面のフィールド 156
- その他のオプションの設定 159
- グローバルパラメータの設定 167

設定オプションは、アンロード／ロード／検索／スキャン機能のオプション設定、またはアンロード／ロード機能のパラメータ設定を指定するために使用します。

手順 28.1. ［Unload/Load/Scan Settings］画面を表示するには

■ アンロード、ロード、またはスキャンの任意の画面で、次の内部コマンドを入力します。

```
SETTINGS
```

「*ダイレクトコマンド*」セクションの「*ナビゲーションと特別な機能のためのコマンド*」も参照してください。

または:

上級ユーザーモードをアクティブにし、機能を選択し ENTER キーを押して、処理プロシージャを開始します。

または:

上級ユーザー画面で PF7 キー（Setti）を押します。

デフォルトで選択されていない場合に、以下で説明する［**Unload/Load/Scan Settings**］画面に表示されるオプションをアクティブにするには、対応する入力フィールドを任意の単一文字でマークします。

設定画面のフィールド

［**Unload/Load/Scan Settings**］画面には、次のフィールドおよび PF キーが表示されます。

フィールド	PF キー	説明
Transfer format		有効になるのは、［**Use default options**］（これがデフォルトです）を選択した場合に限られます。 これを選択すると、ワークファイルにおける処理データの書き込み／読み取りが転送フォーマットで実行されます。「ワークファイル」の「*ワークファイルフォーマット*」も参照してください。 アンロード機能： アンロードするデータは、転送フォーマットでワークファイルに書き込まれます。以降のアンロードではこのフィールドの設定を変更する場合は、［**Main Menu**］に戻るか、コマンド GO UNLOAD END（「*ダイレクトコマンド*」の「*ナビゲーションと特別な機能のためのコマンド*」を参照）を入力してアンロード機能を再スタートする必要があります。 ロードおよびスキャン機能：

フィールド	PF キー	説明
		ロードまたはスキャンするデータは転送フォーマットにする必要があります。
Unicode work file		アンロード機能にのみ適用され、有効になるのは［**Transfer format**］が選択されている場合に限られます。 このオプションが選択されている場合、すべてのオブジェクトソースが、ワークファイルに書き込まれる前に Unicode/UTF-8（Universal Transformation Format、8 ビット形式）に変換されます。 Unicode ワークファイルが指定されている場合、転送オプション［**Use conversion table**］、［**Substitute line references**］、および［**Incorporate free rules**］は使用できません。
Use PC File		適用されるのは、Entire Connection がインストールされている場合に限られます。 これを選択すると、処理データが Entire Connection ワークファイルから読み込まれるか、このワークファイルに書き込まれます。
Portable work file		ロード／スキャン機能を使用する場合には設定する必要はありません。これらの機能では自動的に適切なワークファイルタイプが選択され、このオプションは設定されていても無視されます。 ［**Portable work file**］が有効になるのは、次の条件に該当した場合に限られます。 ■ ［**Use default options**］（これがデフォルトです）が選択されており、 ■ ［**Transfer format**］が選択されていない場合。 ［**Portable work file**］を選択すると、ワークファイルの書き込み／読み取りはポータブルフォーマットで実行されます。「ワークファイル」の「*ワークファイルフォーマット*」も参照してください。
Work file	PF11 PF1	有効になるのは、**Use default options**（これがデフォルトです）を選択した場合に限られます。 ■ その機能を使用するワークファイルの名前。名前が使用可能なスペースを超える場合、PF11 キー（WorkF）を押すと名前をさらに長く入力できます（最大 253 文字）。または、このフィールドでカーソルを表示して PF1（ヘルプ）キーを押します。 「*ワークファイル*」を参照してください。 ■ または、**Use PC File** が選択されている場合、パス名および Entire Connection ワークファイルが使用されます。 「*ワークファイル*」を参照してください。
Use default options		デフォルトオプションが使用されます（これがデフォルトです）。「*プロファイルの設定*」および「*その他のオプションの設定*」も参照してください。

フィールド	PF キー	説明
Set additional options	PF4	有効になるのは、［**Use default options**］を選択した場合に限られます。 処理シーケンスのデフォルト設定を修正し、追加オプションを入力できる［**Options**］画面が表示されます。使用可能なオプションについては、「*その他のオプションの設定*」を参照してください。
Use Option Workplan		タイプ OPTION のワークプランが使用されます。「*ワークプラン*」も参照してください。
Name （［Use Option Workplan］の横）		有効になるのは、［**Use Option Workplan**］を選択した場合に限られます。 使用するタイプ OPTION のワークプランの名前。
List Option Workplan	PF6	有効になるのは、［**Use Option Workplan**］を選択した場合に限られます。 指定したワークプランの内容が［**Use Option Workplan**］の横にある［**Name**］フィールドに表示されます。
Select Option Workplan	PF5	有効になるのは、［**Use Option Workplan**］を選択した場合に限られます。 タイプ OPTION の使用可能なワークプランの選択リストが表示されます（「*管理*」の「*ワークプランライブラリ内の利用可能なワークプランのリストの作成*」も参照）。
Do not use parameters		これが選択されている場合（デフォルト設定）、パラメータは設定されません。
Use global parameters		グローバルパラメータが使用されます。「*グローバルパラメータの設定*」を参照してください。
Set global parameters	PF7	有効になるのは、［**Use global parameters**］を選択した場合に限られます。 ［**Parameters**］画面が表示されます。キーワードと有効な入力値の詳細については、「*グローバルパラメータの設定*」および「*`parameter-setting`*」（「*ダイレクトコマンド*」）を参照してください。
Use Parameter Workplan		タイプ PARAMETER のワークプランが使用されます。「*ワークプラン*」も参照してください。
Name （［Use Parameter Workplan］の横）		有効になるのは、［**Use Parameter Workplan**］を選択した場合に限られます。 使用するタイプ PARAMETER のワークプランの名前。
List Parameter Workplan	PF9	有効になるのは、［**Use Parameter Workplan**］を選択した場合に限られます。 指定したワークプランの内容が［**Use Parameter Workplan**］の横にある［**Name**］フィールドに表示されます。
Select Parameter Workplan		有効になるのは、［**Use Parameter Workplan**］を選択した場合に限られます。 タイプ PARAMETER の使用可能なワークプランの選択リストが表示されます。「*管理*」の「*ワークプランライブラリ内の利用可能なワークプランのリストの作成*」も参照してください。

その他のオプションの設定

［**Options**］画面に表示されるセクションについて、以下で説明します。すべてのセクションが画面に表示されるわけではありません。これは、使用する機能、定義された設定、およびインストールされている製品によって、セクションが異なるからです。

キーワードと有効な入力値の詳細については、「*ダイレクトコマンド*」セクションの `option-setting` に関する記載も参照してください。

このセクションでは、次のトピックについて説明します。

- ワークファイルオプションとレポートオプション
- XREF オプション
- XRef について
- 転送オプション
- 置換オプション
- 処理するオブジェクト数
- FDIC 設定
- FSEC 設定

ワークファイルオプションとレポートオプション

次のセクションでは、ワークファイルおよびレポート用に表示されるオプションについて説明します。

フィールド	説明
Use PC File	適用されるのは、Entire Connection がインストールされている場合に限られます。 これを選択すると、処理データが Entire Connection ワークファイルから読み込まれるか、このワークファイルに書き込まれます。
Work file	有効になるのは、**Use default options**（これがデフォルトです）を選択した場合に限られます。 ■ その機能を使用するワークファイルの名前。名前が使用可能なスペースを超える場合、PF11 キー（WorkF）を押すと名前をさらに長く入力できます（最大 253 文字）。または、このフィールドでカーソルを表示して PF1（ヘルプ）キーを押します。 「*ワークファイル*」を参照してください。 ■ または、**Use PC File** が選択されている場合、パス名および Entire Connection ワークファイルが使用されます。 「*ワークファイル*」を参照してください。

フィールド	説明
Unicode work file	アンロード機能にのみ適用され、有効になるのは［**Transfer format**］が選択されている場合に限られます。 このオプションが選択されている場合、すべてのオブジェクトソースが、ワークファイルに書き込まれる前に Unicode/UTF-8（Universal Transformation Format、8 ビット形式）に変換されます。 Unicode ワークファイルが指定されている場合、転送オプション［**Use conversion table**］、［**Substitute line references**］、および［**Incorporate free rules**］は使用できません。
Write report	処理するオブジェクトのレポートをワークファイル 4 に書き込みます。 オプション［**Write report**］は、デフォルトで選択されています。 レポートを表示するには、内部コマンド `SHOW REPORT FILE` を入力します（「*ダイレクトコマンド*」の「*ナビゲーションと特別な機能のためのコマンド*」を参照）。
Start new report	有効になるのは、［**Write report**］を選択した場合に限られます。 新しいレポートが書き込まれる前に、ワークファイル 4の内容を削除します。この指定がないと、新しいレポートが既存のレポートに付加されます。
Error report only	有効になるのは、［**Write report**］を選択した場合に限られます。 エラーメッセージのみをレポートに書き込みます。これには、Natural Security からのメッセージ、および LOAD コマンドの実行中に発生したメッセージ（例：「置換されませんでした」）が含まれます。「*ダイレクトコマンド*」、「*option-setting*」の「`REPORT-OPTION-1`」も参照してください。
Report file	有効になるのは、［**Write report**］を選択した場合に限られます。 レポートファイルの名前：ワークファイル4に割り当てられた完全パス名を入力します。 名前が使用可能なスペースを超える場合、PF5（RepoF）キーを押すと名前をさらに長く入力できます（最大253 文字）。または、このフィールドでカーソルを表示して PF1 キー（ヘルプ）を押します。
Write restart information	ロード機能にのみ適用されます。 このオプションを設定すると、ロードの再スタート機能に対して再スタート情報が提供されます。 詳細については、「*機能*」の「*ロードの再スタート*」を参照してください。
Restart file	［**Write restart information**］が選択された場合に、ロード機能にのみ適用されます。 再スタートデータに使用するワークファイルの名前：ワークファイル 6（デフォルト設定）または指定した `restart-file`。 名前が使用可能なスペースを超える場合、PF6（RestF）キーを押すと名前をさらに長く入力できます（最大253 文字）。または、このフィールドでカーソルを表示して PF1 キー（ヘルプ）を押します。

フィールド	説明
Use FDDM file for processing DDMs	NATPARM パラメータファイルで FDDM システムファイルが有効になっている環境でのみ適用されます。 このオプションを選択する（これがデフォルトです）と、ロード、アンロード、または検索機能で DDM の処理に FDDM システムファイルが使用されます。 処理する DDM に対して、ライブラリ `SYSTEM` と Natural オブジェクトタイプ `V` を指定します（「*オブジェクト指定*」の「***Natural** ライブラリオブジェクトの詳細*」を参照）。 ロード機能でこれを使用した場合、すべての DDM が FDDM システムファイルにロードされます。この場合、パラメータ `NEWLIBRARY` は無視されます。 「*ダイレクトコマンド*」の *`option-clause`* の構文図も参照してください。
Delete allowed	ロード機能にのみ適用されます。オブジェクトを内部フォーマットでロードするときに、ワークファイルから削除命令を処理します。

XREF オプション

XREF オプションは、データを内部フォーマットでアンロードまたはロードする場合（つまり、［**Transfer format**］フィールドが選択されていない場合）にのみ使用できます。XRef データを処理するには、Predict がインストールされている必要があります。

次のセクションでは、表示される XREF オプションおよびそれが適用される機能について説明します。

フィールド	説明	関数
Yes (unload XRef data) または **Yes (load XRef data)**	カタログ化オブジェクトとそのクロスリファレンスデータ（存在する場合）をアンロードします。クロスリファレンスがワークファイルに存在する場合は、カタログ化オブジェクトとそのクロスリファレンスデータをロードします。	アンロード ロード
No (ignore XRef data)	XRef データは処理されません。	アンロード ロード
Force	カタログ化オブジェクトとそのクロスリファレンスデータをロードします。ただしそれは、クロスリファレンスがワークファイル内に存在し、そのオブジェクトの Predict エントリが FDIC システムファイル内に存在する場合に限られます。	ロード
Doc	カタログ化オブジェクトとそのクロスリファレンスデータ（存在する場合）をロードします。ただしそれは、そのオブジェクトの Predict エントリが FDIC システムファイル内に存在する場合に限られます。	ロード
Special	カタログ化オブジェクトとそのクロスリファレンスデータ（存在する場合）をロードします。	ロード

XRef について

Predict システムファイルに格納されたすべてのクロスリファレンス（XRef）データは、オブジェクトハンドラで処理できます。XREF オプションでは、XRef データをオブジェクトハンドラで処理する必要があるかどうかが示されます。カタログ化オブジェクトに削除または置き換え機能が実行される場合、XRef データは常に削除されます。

Predict がインストールされていない場合、XREF オプションを N に設定すると、Predict ファイルの検証は実行されません。XREF オプションを Y に設定したときに、使用中の FDIC ファイルが有効な Predict ファイルでない場合、エラーメッセージが返されます。

XREF オプションを設定する場合のルールは、Natural Security によるルールと同じです。非セキュリティ環境には制限はありません。以下の最初の 5 つのケースを参照してください。ただし、Natural Security がアクティブな場合、前回のケースと同様に、オブジェクトハンドラの XREF オプションの設定は、Natural Security のユーティリティプロファイルの XREF オプションの値によって異なります。

XREF が次のように設定されている場合を考えます。

- XREF が OFF または No に設定されている場合
- XREF が ON または Yes あるいは Force に設定されている場合
- XREF が Force に設定されている場合
- XREF が Doc に設定されている場合
- XREF が Special に設定されている場合
- Natural Security でのXREF オプション

XREF が OFF または No に設定されている場合

XREF オプションが **OFF** または **No** に設定されている場合、XRef データは処理されません。ただし、カタログ化されたオブジェクトが削除または置換された場合、オブジェクトハンドラは XRef データを削除します。ターゲット Predict システムファイルは、FDIC オプションの現在の設定に従って決定されます。デフォルトでは、Natural セッションの開始時に FDIC プロファイルパラメータに割り当てられた値です（『パラメータリファレンス』ドキュメントの「*FDIC - Predict システムファイル*」を参照）。

XREF が ON または Yes あるいは Force に設定されている場合

XREF オプションを **Yes** または **Force** に設定した場合、処理中に次のアクションが適用されます。

- アンロード
  カタログ化オブジェクトとそのクロスリファレンスデータ（存在する場合）をアンロードします。

- ロード
  クロスリファレンスがワークファイルに存在する場合は、カタログ化オブジェクトとそのクロスリファレンスデータをロードします。

XREF が Force に設定されている場合

LOAD にのみ適用されます。

カタログ化オブジェクトとそのクロスリファレンスデータをロードします。ただしそれは、クロスリファレンスがワークファイル内に存在し、そのオブジェクトの Predict エントリが FDIC システムファイル内に存在する場合に限られます。

XREF オプションが **Force** に設定されている場合、オブジェクトハンドラでは、Predict システムターゲットファイルで定義された Predict プログラムエントリがあるかどうかがさらにチェックされます。ない場合、オブジェクトの処理は終了します。

XREF が Doc に設定されている場合

LOAD にのみ適用されます。

XREF オプションが **Doc** に設定されている場合、オブジェクトハンドラでは、Predict システムターゲットファイルで定義された Predict プログラムエントリがあるかどうかがチェックされます。ない場合、オブジェクトの処理は終了します。クロスリファレンスデータを含むカタログ化オブジェクトはそのクロスリファレンスデータで処理され、クロスリファレンスデータを含まないカタログ化オブジェクトも処理されます。

XREF が Special に設定されている場合

LOAD にのみ適用されます。

XREF オプションを **Special** に設定した場合、特殊ケースが適用され、指定したカタログ化オブジェクトの範囲は、すべてのカタログ化オブジェクトにクロスリファレンスデータがあるかどうかに関係なく、対応する XRef データとともに処理されます。クロスリファレンスデータがあるカタログ化オブジェクトはクロスリファレンスデータとともに処理され、クロスリファレンスデータがないカタログ化オブジェクトも処理されます。

Natural Security でのXREF オプション

Natural Security がアクティブの場合、次のルールが適用されます。

- Natural Security のユーティリティプロファイルにおいて XREF オプションの値が `N` または `S` の場合、オブジェクトハンドラで XREF オプションに対して任意の値（**OFF/No**、**ON/Yes**、**Doc**、**Force** または **Special**）を指定できます。
- Natural Security のユーティリティプロファイルにおいて XREF オプションの値が `D` の場合、オブジェクトハンドラで XREF オプションに対して指定できる値は **Doc** または **Force** のみで

す。**ON/Yes** を指定した場合、値 **Force** が内部的に使用されます。**OFF/No** または **Special** を指定した場合、値 **Doc** が内部的に使用されます。

- Natural Security のユーティリティプロファイルにおいて XREF オプションの値が Y の場合、オブジェクトハンドラで XREF オプションに対して指定できる値は **ON/Yes** または **Force** のみです。**Doc** を指定した場合、値 **Force** が内部的に使用され、**OFF/No** または **Special** を指定した場合、値 **ON/Yes** が内部的に使用されます。
- Natural Security のユーティリティプロファイルにおいて XREF オプションの値が F である場合、オブジェクトハンドラで XREF オプションに対して指定できる値は **Force** のみです。他の値を指定した場合、値 **Force** が内部的に使用されます。

転送オプション

転送オプションは、データを内部フォーマットで処理している場合（つまり、［**Transfer format**］フィールドが選択されている場合）にのみ使用できます。

次のセクションでは、表示される転送オプションおよびそれが適用される機能について説明します。

オプション	説明	関数
Substitute line references	適用されるのは、ソースコードの行番号がステートメント参照に使用されている場合に限られます。 行番号がソースコードの参照として使用されている場合、参照先の行の行番号と行番号参照はラベルで置き換えられます。ソースはデータベースでは変更されません。	アンロード
Include line numbers	このオプションを選択すると、行番号が転送されます。（デフォルトでは、Natural オブジェクトの行番号は転送されません。）	アンロード
Incorporate free rules	Predict がインストールされている場合、マップと関連付けられた Predict ルールがマップソースに組み込まれます。	アンロード
Use conversion table	注意： このオプションは、異なる文字セットを使用する環境間で非 FTP 転送を実行しており、転送ツールによって変換が行われないという特殊な場合にのみ使用します。 アンロード： 内部の Natural 変換テーブル（「**System table**」参照）またはユーザーによって定義された変換テーブル（「**User table**」参照）を使用して、データを EBCDIC フォーマットに変換します。 ロード： 内部の Natural 変換テーブル（「**System table**」参照）またはユーザーによって定義された変換テーブル（「**User table**」参照）を使用して、データを ASCII フォーマットに変換します。ワークファイル内のデータが EBCDIC フォーマットの場合、または変換プログラム（「**User table**」を参照）が指定されている場合にのみ適用されることに注意してください。	アンロード ロード

オプション	説明	関数
System table	有効になるのは、［**Use conversion table**］を選択した場合に限られます。 アンロード： 内部的なNatural変換テーブルを使用して、データをEBCDICフォーマットに変換します。 ロード： 内部的なNatural変換テーブルを使用して、データをASCIIフォーマットに変換します。	アンロード ロード
User table	有効になるのは、［**Use conversion table**］を選択した場合に限られます。 変換プログラム名をフィールドに入力すると、定義した変換プログラムを使用して、データが EBCDIC または ASCII フォーマットに変換されます。個々の変換プログラムを指定するには、ライブラリ SYSOBJH またはその steplib の 1 つにプログラムが保存されている必要があります。ライブラリ SYSOBJH にあるサブプログラム例 OTNCONAE および OTNCONEA を参照してください。 変換プログラムが指定されていない場合、デフォルトで Natural ファイル *NATCONV.INI* 内のこれに相当する変換テーブルがアンロード（[ISO8859_1->EBCDIC]）機能およびロード（[EBCDIC->ISO8859_1]）機能で使用されます。	アンロード ロード
Use load code page	このオプションを選択すると、ロード機能に使用するコードページ名を入力できるウィンドウが表示されます。 このオプションを選択すると、UTF-8でワークファイルにアンロードされたすべてのオブジェクトソースは、ワークファイルにロードされるときに指定のコードページで変換されます。**Unicode work file** も参照してください。 コードページ名として「`*CODEPAGE`」を入力すると、システム変数`*CODEPAGE`に割り当てられている値が使用されます（『システム変数』ドキュメントを参照）。 コードページ名を指定しないと、ソースオブジェクトは、アンロードするときに使用したコードページで変換されます。 ［**Use load code page**］が指定されている場合、［**Use conversion table**］オプションと［**Translate to upper case**］オプションは使用できません。	ロード
Translate to upper case	ロードするソースコードを大文字に変換します。	ロード
Data area format	データエリアにのみ適用されます。 データエリアソースをアンロードまたはロードするフォーマットを指定します。使用可能な入力値は次のとおりです。 `N` データエリアを新しい内部データエリアフォーマットに変換します。	アンロード ロード

オプション	説明		関数
	0	データエリアを以前の内部データエリアフォーマットに変換します。1つ以上のデータエリアソースを以前の内部データエリアフォーマットに変換できない場合、アンロードの完了時にオブジェクトハンドラが該当するメッセージを発行します。さらに、アンロード機能によって生成されたアンロードレポートの［**Status**］列（影響を受けたデータエリアソース名の横）に、該当する注釈が表示されます。	
	*	データエリアを変換しません。これがデフォルトです。	
	詳細については、『*エディタ*』ドキュメントの「*データエリアエディタ*」を参照してください。		

置換オプション

次に説明する置換オプションは、ロード機能にのみ適用されます。

Do not replace	オブジェクトを置換しません。これがデフォルトです。
Replace all	すべてのオブジェクトを置換します。
Replace obsolete	ロードファイル内のオブジェクトの日付より前の日付になっているオブジェクトを置換します。
Replace except newer	ロードファイル内のオブジェクトの日付より新しい日付になっているオブジェクトを除いた、すべてのオブジェクトを置換します。

処理するオブジェクト数

処理するオブジェクト数は、ロード機能およびスキャン機能にのみ適用されます。

［**Number to process**］フィールドに、最大 5 桁の値を入力します。0 より大きい値が指定された場合、指定した数のオブジェクトを処理した後、ロードまたはスキャン機能は停止します。

注意: カタログされた Natural オブジェクトが同じ名前のソースオブジェクトの直後に処理された場合、両者で 1 つのオブジェクトと見なされます。

FDIC 設定

FDIC 設定は、Predict がインストールされている場合にのみ適用されます。

XRef データの処理に使用する Predict ファイル（FDIC）を指定します。

DBID	FDIC ファイルが保存されているデータベースの ID。
FNR	FDIC ファイルが保存されているファイルの番号。
Password	指定は任意です。 FDIC ファイルが保存されている Adabas ファイルの Adabas パスワード。
Cipher	指定は任意です。 FDIC ファイルが保存されている Adabas ファイルのサイファコード。

FSEC 設定

FSEC 設定は、Natural Security がインストールされている場合にのみ適用されます。

FSEC 設定は、セキュリティチェックに使用する Natural Security データファイル（FSEC）を指定するために使用されます。

DBID	FSEC ファイルが保存されているデータベースの ID。
FNR	FSEC ファイルが保存されているファイルの番号。
Password	指定は任意です。 FSEC ファイルが保存されている Adabas ファイルの Adabas パスワード。
Cipher	指定は任意です。 FSEC ファイルが保存されている Adabas ファイルのサイファコード。

グローバルパラメータの設定

ロード機能またはアンロード機能にのみ適用されます。

［**Parameters**］画面に表示されるフィールドは、ロードまたはアンロード機能で処理するオブジェクトのグローバルパラメータの設定を変更したり、ロード機能のターゲット環境を変更したりするために使用できます。例えば、選択したオブジェクトをワークファイルにアンロードするために新しい名前（または名前の範囲）を指定したり、選択したオブジェクトをワークファイルからロードするために別のライブラリを指定したりすることができます。

アンロード機能でグローバルパラメータが指定されている場合、このパラメータ設定はワークファイルに書き込まれる前のオブジェクトに影響を与えます。ロード機能でグローバルパラメータが指定されている場合、このパラメータ設定はターゲット環境に書き込まれる前のオブジェクトに影響を与えます。

パラメータ設定を変更するために指定できる値は、目的のパラメータの横にある［**Check Value**］および［**New Value**］フィールドに入力されます。

［**Check Value**］に値が入力されなかった場合、［**New Value**］に入力された値は、特定のパラメータ設定が適用されるすべてのオブジェクトに影響します。［**Check Value**］に値が入力された場合、［**New Value**］に入力された値は、特定のパラメータ設定と［**Check Value**］に入力された値が適用されるオブジェクトにのみ影響します。［**Check Value**］または［**New Value**］が処理するオブジェクトタイプに関連していない場合、どちらのフィールドに入力された値も無視されます。例えば、次のようになります。Naturalシステムエラーメッセージにライブラリ名は含まれていません。したがって、Naturalシステムエラーメッセージを処理するとき、［**Library**］フィールドの［**Check Value**］または［**New Value**］に入力された値は無視されます。

［**Check Value**］および **New Value** は、［**Parameters**］画面の［**System files for load**］セクションに含まれるパラメータおよびパラメータ **Error number difference** には適用されません。

有効なパラメータ設定については、「*ダイレクトコマンド*」セクションの「*parameter-setting*」も参照してください。

［**Parameters**］画面には、次のフィールドが含まれます。

フィールド／セクション	説明
Object name	**Check Value/New Value:** 単一のオブジェクト名または名前の範囲。「*名前と日時の指定*」の「*名前*」、および「*新しい値のルール*」を参照してください。 **注意:** メインフレームプラットフォーム上の DDM には適用されません。
Library	**Check Value/New Value:** 単一のライブラリ名または名前の範囲。「*名前*」および「*新しい値のルール*」を参照してください。
Date	**Check Value/New Value:** 単一の日付または日付の範囲。「*名前と日時の指定*」の「*日付*」、および「*新しい値のルール*」を参照してください。
Time	**Check Value/New Value:** 単一の時刻または時刻の範囲。「*名前と日時の指定*」の「*時刻*」、および「*新しい値のルール*」を参照してください。
User ID	**Check Value/New Value:** 単一のユーザー ID またはユーザー ID の範囲。「*名前*」および「*新しい値のルール*」を参照してください。

フィールド／セクション	説明
Lang. codes	適用されるのは、Natural システムエラーメッセージまたはユーザー定義エラーメッセージを処理する場合に限られます。 **Check Value/New Value:** 有効な言語コードは最大 8 つまでです（例えばスペイン語の場合はコード 4）。複数の言語コードを指定する場合、［**Check Value**］に同じ数の言語コードが含まれている必要があります。この場合、［**Check Value**］の言語コードは、対応する［**New Value**］の言語コードで置き換えられます。 注：英語（コード 1）が唯一の有効な言語である長文テキストの Natural システムエラーメッセージには、［**New Value**］は適用されません。
Error number difference	適用されるのは、Natural システムエラーメッセージまたはユーザー定義エラーメッセージを処理する場合に限られます。 エラーメッセージの新規の番号の範囲として使用される 4 桁の正または負の値（+/-*nnnn*）。選択したエラーメッセージに新しい範囲を適用できるかどうかを確認するため、［**Error number from/to** ］フィールド（「*Natural ライブラリオブジェクト*」を参照）に開始値と終了値を指定する必要があります。 例： ［**Error number from/to**］でメッセージ番号 1～10 を選択し、［**Error number difference**］が 2000 に設定されていると、メッセージ番号が 2001～2010 に再設定されます。［**Error number difference**］の値を -1000 に設定すると、検証エラーが発生します。
FDT DBID/FNR	**Check Value/New Value:** Adabas FDT の有効なデータベース ID（DBID）およびファイル番号（FNR）のいずれかあるいはその両方。
Ext. Path	**Check Value/New Value:** 外部オブジェクトのパス名。 名前が使用可能なスペースを超える場合、**Check Value** に対しては PF6 キー（CPath）、**New Value** に対しては PF7 キー（NPath）を押すと名前をさらに長く入力できます（最大 253 文字）。または、いずれかのフィールドにカーソルを置き、PF1 キー（ヘルプ）を押します。
ロード用のシステムファイル：Load FNAT **DBID** **FNR**	ロード機能にのみ適用されます。 ターゲット FNAT システムファイルのデータベース ID（**DBID**）とファイル番号（**FNR**）。このシステムファイルは、ライブラリ名が SYSTEM ではない SYS で始まるすべてのライブラリオブジェクトに対して使用されます。

フィールド／セクション	説明
ロード用のシステムファイル：Load FUSER **DBID** **FNR**	ロード機能にのみ適用されます。 ターゲット FUSER システムファイルのデータベース ID（**DBID**）とファイル番号（**FNR**）。このシステムファイルは、ライブラリ名が SYS で始まらないすべてのライブラリオブジェクトと、ライブラリ SYSTEM に対して使用されます。
ロード用のシステムファイル：Load FNAT/FUSER **Select**	ロード機能にのみ適用されます。 ［**Select System File**］ウィンドウが開き、Natural 環境で使用可能なすべてのシステムファイルがリスト表示されます。「*システムファイルの選択*」を参照してください。
ロード用のシステムファイル：Load NCP **DBID** **FNR**	ロード機能にのみ適用されます。 Natural コマンドプロセッサソースをロードするターゲット Adabas ファイルのデータベース ID（［**DBID**］）とファイル番号（［**FNR**］）。

このセクションでは、次のトピックについて説明します。

- 新しい値のルール

新しい値のルール

Object name、**Library**、**Date/Time** および **User ID** の **New Value** には以下が適用されます。

［**New Value**］にアスタリスク（*）付きの範囲（ABC* など）が含まれている場合、アスタリスク（*）の前の文字数によって、［**Check Value**］で置き換えられる文字数が決まります。これは、［**Check Value**］が［**New Value**］で指定した範囲より短い場合にも有効です（以下の*例*を参照）。

例：

1. ［**Object name**］が ABCDEFG で、［**New Value**］が ZYX* に設定されると、結果のオブジェクト名は ZYXDEFG になります。
2. ［**Object name**］が AB で、［**New Value**］が ZYX* に設定されると、結果のオブジェクト名は ZYX になります。
3. ［**Date/Time**］が 2005-03-26 で、［**New Value**］が 2006* に設定されると、結果のオブジェクト日付は 2006-03-26 になります。

29 「ワークプラン」

- ワークプランの作成、選択、および変更 172
- ワークプランの内容 172
- ワークプランの例 173
- ワークプランの参照 174

ワークプランは、コマンド実行、オブジェクト選択、およびパラメータやオプション設定を行うための個々の標準プロシージャを定義します。ワークプランを使用することで、機能処理をさらに自動化することができます。

ワークプランはテキストタイプのNaturalオブジェクトです。ワークプランは、デフォルトにより、現在の FUSER システムファイルの WORKPLAN ライブラリに保存されます。

ワークプランの作成、選択、および変更

管理機能（関連するセクションを参照）を使用して、ワークプランの作成、リストからのワークプランの選択、ワークプランの変更、およびワークプランのデフォルトライブラリの変更を行うことができます。オブジェクトハンドラプロファイルで`Workplan-Library`パラメータを指定して、デフォルトライブラリを変更することもできます（「*プロファイルの設定*」を参照）。

ワークプランの内容

ワークプランは、（オブジェクトハンドラによって生成された）ヘッダーおよび関連する命令やテキストの部分で構成されます。命令部分にはオブジェクトハンドラコマンドおよびパラメータやオプション設定が含まれます。テキスト部分には単純なテキストだけが含まれます。ヘッダーと命令またはテキスト部分には、コメント（例：ワークプランの簡単な説明）を含むことができます。コメントは必ずデリミタ文字 `/*` で始まり、1 行に制限されています。

ワークプランには、次の 6 つのタイプがあります。PROCEDURE、SELECTION、LIST、PARAMETER、OPTION および TEXT

下の表は、対応するワークプランタイプの有効なヘッダー（オブジェクトハンドラの外部でワークプランを作成する場合に入力される）のリストと命令またはテキスト部分の内容の説明です。また、オブジェクトハンドラのダイレクトコマンドを指定するときに適用される節へのクロスリファレンスも示します。ここで示すオブジェクトハンドラのダイレクトコマンドについては、「*ダイレクトコマンド*」セクションを参照してください。

有効なヘッダー	Contents	「*ダイレクトコマンド*」の関連トピック
`TYPE PROCEDURE`	オブジェクトハンドラコマンドプロシージャの開始。 このワークプランには、PROCEDURE で使用できるオブジェクトハンドラコマンドの任意の組み合わせを含めることができます。セミコロン（;）で区切られた一連のコマンドを入力します。	*基本的なコマンド構文*
`TYPE SELECTION`	オブジェクトの選択条件。	*`select-clause`*

有効なヘッダー	Contents	「ダイレクトコマンド」の関連トピック
	このワークプランは、オブジェクトハンドラのワークプランコマンドで使用できます。	
TYPE LIST	オブジェクトのリスト。 このワークプランは、オブジェクトハンドラのワークプランコマンドで使用できます。	*select-clause* *オブジェクトリスト - LIST ワークプラン*
TYPE PARAMETER	アンロードまたはロード機能のパラメータ。 このワークプランは、オブジェクトがロードされる新しいターゲットライブラリの名前など、処理するオブジェクトの属性を変更するために使用できます。 TYPE PARAMETER は、オブジェクトハンドラのワークプランコマンドで使用できます。	*parameter-setting*
TYPE OPTION	アンロードまたはロード機能のオプション（レポート設定など）。 このワークプランは、オブジェクトハンドラのワークプランコマンドで使用できます。	*option-setting*
TYPE TEXT	ドキュメントのために使用できる任意のコメントテキスト。	該当なし

ワークプランの例

次の表に、ワークプランに含まれる命令部分の例を示します。

ワークプランのタイプ	命令	説明
PROCEDURE	FINDLIB * LIB TEST	ライブラリ TEST の存在をチェックします。
PROCEDURE	UNLOAD A* LIB TEST	A で始まるすべての Natural プログラミングオブジェクトと共有リソース、およびすべてのユーザー定義エラーメッセージをライブラリ TEST からワークファイル 1 にアンロードし、レポートをワークファイル 4 に書き込みます。
SELECTION	* LIB TEST	ライブラリ TEST のすべてのオブジェクトを処理します。
TEXT	This is a Workplan comment.	任意のテキスト。

このセクションでは、次のトピックについて説明します。

- ワークプランの内容の例

ワークプランの内容の例

次に、UNLOAD コマンドを実行したときの PROCEDURE ワークプランの例を示します。

```
TYPE PROCEDURE /* VERSION=03.01 NATURAL VERSION=06.93.09  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
```

ワークプランの参照

オブジェクトハンドラのメニュー機能またはダイレクトコマンド（「*ダイレクトコマンド*」セクションも参照）を使用して、ワークプランを参照することができます。

次の構文は、「*ダイレクトコマンド*」セクションで説明しているオブジェクトハンドラのダイレクトコマンドを使用してワークプランを参照する場合に適用されます。

```
( workplan-name
  [ LIBRARY library-name ]
  [DBID dbid [FNR fnr ]] [NAME vsam-name ]
  [CIPHER cipher ]
  [ { PASSWORD } password ]
  [ { PSW }                ]
)
```

構文オプションについては、次のセクションで説明します。

- キーワードの説明

キーワードの説明

下の表は、ワークプランを参照するための構文に適用されるキーワードと値の説明です。

キーワード	値	デフォルト値
workplan-name	ワークプランとして使用する、ワークプランライブラリに保存された Natural テキストオブジェクトの名前。	デフォルトなし
LIBRARY	ワークプランが保存されているライブラリの名前。	WORKPLAN
DBID	ワークプランライブラリが保存されている Adabas データベースの ID。	0 （現在の FNAT/FUSER）
FNR	ワークプランライブラリが保存されている Adabas ファイルの番号。	0 （現在の FNAT/FUSER）
NAME	メインフレーム上のオブジェクトにのみ適用されます。 ワークプランライブラリが保存されている有効な VSAM ファイルの名前。	空白 （現在の FNAT/FUSER）
CIPHER	メインフレーム上のオブジェクトにのみ適用されます。 8 桁のサイファコード。	空白 （現在の FNAT/FUSER）
PASSWORD	メインフレーム上のオブジェクトにのみ適用されます。 8 文字の Adabas パスワード。	空白 （現在の FNAT/FUSER）

30 名前と日時の指定

- Name 178
- 日付 179
- 時刻 180

単一の名前／日付／時刻、または名前／日付／時刻の範囲を使用して、Naturalライブラリオブジェクト、Natural コマンドプロセッサソース、Natural 関連オブジェクト外部ファイル（外部オブジェクト）を選択できます。

Name

単一の名前または名前の範囲を指定できます。

下記のオプションのリストで、*value* は 1 つ以上の文字の任意の組み合わせを表します。

	入力	選択される項目
	value	*value* に一致する名前を持つすべての項目。
	* >	すべての項目。
	?	入力された疑問符（?）の個数に相当する任意の文字数のすべての項目。
先頭文字	*value**	*value* で始まる名前を持つすべての項目。 例：AB* 選択対象：AB、AB1、ABC、ABEZ 選択対象外：AA1、ACB
ワイルドカード	*value*?	*value* で始まり、入力された疑問符（?）の個数に相当する任意の文字数がその後に続く名前の、すべての項目。 例：ABC? 選択対象：ABCA、ABCZ 選択対象外：AXC、ABCAA
	value?*value*?	任意の順序でアスタリスク（*）と疑問符（?）を指定した *value* に一致するすべての項目。 例：A?C*Z 選択対象：ABCZ、AXCBBBZ、ANCZ 選択対象外：ACBZ、ABDEZ、AXCBBBZA
	*value**value*?	
	value*?*value	
開始値	*value*>	*value* 以降の名前を持つすべての項目。 例：AB> 選択対象：AB、AB1、BBB、ZZZZZZZ 選択対象外：AA1、AAB
終了値	*value*<	*value* 以前の名前を持つすべての項目。 例：AX< 選択対象：AB、AWW、AX 選択対象外：AXA、AY

注意: 先頭文字（アスタリスク（*）の指定）を利用できるのは、パラメータ指定オプション［**New Value**］に限られます。「*設定*」セクションの「*グローバルパラメータの設定*」にある「*新しい値のルール*」を参照してください。

日付

オブジェクトハンドラ内のすべての日付値は国際日付フォーマットで指定されます。

日付、日付の範囲、特殊日付、または特殊日付の範囲を指定できます。日付は、*YYYY-MM-DD*（*YYYY* = 年、*MM* = 月、*DD* = 日）の形式で指定する必要があります。

以下のオプションのリストで、キーワードの下線の部分は有効な省略形を表し、*value* は 1 桁以上の数字の任意の組み合わせを表します。

	入力値	選択される項目
日付	*YYYY-MM-DD*	*YYYY-MM-DD* に等しい日付を持つすべての項目。 例：2003-02-15
先頭文字	*value**	*value* で始まる日付を持つすべての項目。 例：2002* 選択対象：2002-01-01、2002-12-31 選択対象外：2001/12/31、2003/01/01
開始値	*value*>	*value* より大きい日付を持つすべての項目。 例：2002-05> 選択対象：2002-05-01、2002-12-31、2003-01-01、2003-12-31 選択対象外：2002-04-31、2001-12-31 特殊な日付の値を *value* として使用することができます（下記参照）。
終了値	*value*<	*value* より小さい日付を持つすべての項目。 例：2003-02< 選択対象：2002-05-01、2002-12-31、2003-01-01、2003-01-31 選択対象外：2003/02/01、2003/05/18 特殊な日付の値を *value* として使用することができます（下記参照）。
特殊な日付の値		
TODAY (+/-*nnnn*)		当日の日付が含まれるすべての項目。 この後には、+*nnnn* または –*nnnn* を付け加えることができます。*nnnn* は最大 4 桁の数値です。 これを付け加えた場合、現在の日付に *nnnn* 日分を加算または減算した日付が対象として算出されます。

	入力値	選択される項目
		例：当日の日付が 2003-03-01 の場合、TODAY +5 は 2003-03-06 になります。
YESTERDAY		当日の 1 日前の日付が含まれるすべての項目。
MONTH		当月内の日付の範囲が含まれるすべての項目。 例：現在の月は 2003-02 です。 選択対象：2003-02-01、2003-02-30 選択対象外：2003/03/01
		FMDATE：当月の最初の日で開始します。 TODATE：当月の最後の日で終了します。 FMDATE と TODATE の値が同じであれば、選択は 1 日に限定されます。
YEAR		当年内の日付の範囲が含まれるすべての項目。 例：現在の年は 2003 です。 選択対象：2003/01/01、2002/12/31 選択対象外：2002-31-12
		FMDATE：当年の最初の日で開始します。 TODATE：当年の最後の日で終了します。 FMDATE と TODATE の値が同じであれば、選択は 1 年に限定されます。

注意: 先頭文字（アスタリスク（*）の指定）を利用できるのは、パラメータ指定オプション［**New Value**］に限られます。「*設定*」セクションの「*グローバルパラメータの設定*」にある「*新しい値のルール*」を参照してください。

時刻

単一の時刻または時刻の範囲を指定できます。時刻は次の形式で指定する必要があります：*HH*:*II*:*SS*（*HH* = 時間、*II* = 分、*SS* = 秒）

下記のオプションのリストで、*value* は 1 桁以上の数字の任意の組み合わせを表します。

	入力値	選択される項目
時刻	*HH*:*II*:*SS*	*HH*:*II*:*SS* に等しい時刻を持つすべての項目。 例：14:15:16
先頭文字	*value**	*value*. で始まる時刻を持つすべての項目。 例：13:* 選択対象：13:00:00、13:10:53、13:59:59 選択対象外：12:59:59、14:00:00

31 ワークファイル

- ワークファイルの割り当て 182
- ワークファイルフォーマット 183

このセクションでは、オブジェクトハンドラのアンロード／ロード／スキャン機能に適用されるワークファイルおよび有効なフォーマットについて説明します。

「*設定*」セクションの「*ワークファイルオプション*」も参照してください。

注意: ワークファイル名が使用可能なスペースを超える場合、PF11 キーを押すと名前をさらに長く入力できます（最大 253 文字）。または、［**Work file**］フィールドにカーソルを置き PF1（ヘルプ）キーを押します。

ワークファイルの割り当て

下の表は、オブジェクトハンドラで使用されるワークファイルのリストです。

ファイル	説明
ワークファイル 1	アンロード、ロード、およびスキャン機能に使用されます。 アンロードされたデータが含まれます。
ワークファイル 3	内部レポートファイル。
ワークファイル 4	［**Write report**］オプション（「*設定*」の「*ワークファイルオプションとレポートオプション*」を参照）が設定されている場合に使用されます。**Write report** はオブジェクト処理のデフォルト設定です。 レポートデータが含まれます。
ワークファイル 5	ロードする Adabas FDT（フィールド定義テーブル）のためのターゲットファイル。
ワークファイル 6	［**Write restart information**］オプション（「*設定*」の「*その他のオプションの設定*」を参照）が設定されている場合にロード機能で使用されます。 再スタート情報が含まれます。
ワークファイル 7	使用されるのは、Entire Connection がインストールされていて、なおかつ、［**Use PC File**］が［**Options**］画面で選択されている場合に限られます（「*その他のオプションの設定*」も参照）。 アンロード、ロードおよびスキャン機能で使用するには、ワークファイル 7 を Entire Connection ワークファイルとして定義する必要があります。 アンロードされたデータが含まれます。
ワークファイル 8	使用されるのは、Entire Connection がインストールされていて、なおかつ、［**Use PC File**］が［**Options**］画面で選択されている場合に限られます（「*その他のオプションの設定*」も参照）。 アンロード、ロード、およびスキャンの機能で使用するには、ワークファイル 8 を Entire Connection ワークファイルとして定義する必要があります。 Entire Connection コマンドを処理する内部ファイルとして使用されます。

ファイル	説明
	注意: PROCEDURE タイプのワークプランを使用している場合、ユーザー出口ルーチン **OBJHEX03**（「*バッチのコンディションコードとユーザー出口ルーチン*」を参照）またはオプション PCCOMMANDFILENUMBER、PCCOM、および PCCFN（*option-setting* を参照）を使用して、ワークファイルの番号を変更できます。
ワークファイル 9	内部ワークファイル。
ワークファイル 10	トレースワークファイル。トレースモードが設定されている場合に使用されます。 「*ダイレクトコマンド*」の「*ナビゲーションと特別な機能のためのコマンド*」の「SET TRACE WORKFILE」も参照してください。

ワークファイルフォーマット

ソース環境のオブジェクトをワークファイルにアンロードし、それらをワークファイルからターゲット環境にロードするために、2つのファイルフォーマット（内部フォーマットと転送フォーマット）が用意されています。バイナリデータを転送するには、ワークファイルを内部フォーマットにする必要があります。テキストデータを転送するには、ワークファイルを転送フォーマットにする必要があります。

このセクションでは、次のトピックについて説明します。

- 内部フォーマット
- 転送フォーマット

内部フォーマット

内部フォーマットとは、Natural ソースおよびカタログ化オブジェクト、エラーメッセージ、コマンドプロセッサ、Adabas FDT（フィールド定義テーブル）および非 Natural オブジェクトを、1つの環境から別の環境に転送するために使用されるワークファイルの内部レコードレイアウトのことです。

同一プラットフォーム間でオブジェクトを転送するには、内部フォーマットのワークファイルを使用します。異なる UNIX、OpenVMS、または Windows プラットフォーム間でオブジェクトを転送する場合（リトルエンディアンマシンからビッグエンディアンマシンに転送する場合など）には、内部フォーマットのポータブルワークファイルを使用します。「*設定*」セクションの「**Portable work file**」、「*ポータブル Natural 生成プログラム*」（『*プログラミングガイド*』）、および DEFINE WORK FILE（『*ステートメント*』ドキュメント）も参照してください。

オブジェクトハンドラでは、デフォルトで内部フォーマットが使用されます。内部フォーマットを使用する（［**Transfer format**］オプションが選択されていない）場合、ワークファイル 1 をバイナリフォーマットにする必要があります。これを行うには、ファイル拡張子を省略するか、ファイル拡張子 .sag を使用します。

内部フォーマットが有効な場合、Natural オブジェクトのソース環境からの読み取りと Natural ワークファイルへの書き込みは、オブジェクトハンドラのアンロード機能によって行われます。このワークファイルは、標準的なファイル転送サービスで別の環境に転送できます。ターゲット環境では、オブジェクトハンドラのロード機能を使用して、オブジェクトをワークファイルから読み取り、ローカルファイルまたはデータベースシステムにロードすることができます。

注意: ユーティリティ NATUNLD によってサーバーで作成されたワークファイルは、内部フォーマットで処理する必要があります。ワークファイルは、NATUNLD を適用したときと同じプラットフォームのサーバーで作成する必要があります。

転送フォーマット

「*設定*」セクションの「**Transfer format**」も参照してください。

転送フォーマットは、ワークファイルの一般的なレコードレイアウトであり、ロードデータまたはアンロードデータを含みます。このフォーマットはプラットフォームに依存せず、Natural オブジェクトのソース、Natural コマンドプロセッサソース、エラーメッセージ、および Adabas FDT を、1 つのハードウェアプラットフォームから別のハードウェアプラットフォームに転送したり、UNIX、OpenVMS、メインフレームと Windows プラットフォーム間で転送したりするために使用できます。

［**Transfer format**］オプションが設定されていると、オブジェクトハンドラのアンロード機能によって、ハードウェアプラットフォームから Natural オブジェクトが読み取られた後、それらが再構築されます。

フォーマットされたレコードは Natural ワークファイルに書き込まれます。このワークファイルは、標準的なファイル転送サービスで別のプラットフォームに転送できます。ターゲットプラットフォームでは、オブジェクトハンドラのロード機能によってオブジェクトがワークファイルから読み取られ、ローカルファイルまたはデータベースシステムにロードされます。ワークファイルから読み取られたオブジェクトは、新しいハードウェアプラットフォームの構造に従って再構築されます。

ワークファイルの指定

転送フォーマットが指定されている（［**Transfer format**］が設定されている）場合、ワークファイル 1 をテキスト（ASCII）フォーマットにする必要があります。これを行うには、`.sag` 以外のファイル拡張子を使用する必要があります。

Unicode/UTF-8 でのソースの処理

転送フォーマットは、Unicode/UTF-8（Universal Transformation Format、8 ビット形式）で Natural オブジェクトソースをアンロードまたはロードするためにも使用します。該当するアンロードオプションを指定した場合（コマンドモードで `WORKFILETYPE` を `UTF-8` に設定するか、メニューモードで［**Unicode work file**］を選択した場合）、すべてのオブジェクトソースは UTF-8 でワークファイルにアンロードされます。該当するロードオプションを指定した場合（コマンドモードで `LOAD-CODE-PAGE` を指定するか、またはメニューモードで［**Use load code page**］を

選択した場合）、UTF-8 のすべてのオブジェクトソースは、Natural システムファイルにロードされるときに指定のコードページで変換されます。

SYSTRANS によるワークファイル

ユーティリティ SYSTRANS によって作成されたワークファイルは、転送フォーマットで処理します。UTF-8 でエンコードされたオブジェクトソースを含むワークファイルは、SYSTRANS で処理できません。

32 ダイレクトコマンド

オブジェクトハンドラには、次の処理を行うためにダイレクトコマンドが用意されています。

- オブジェクトハンドラメニューを使用しない、バッチモードまたはダイレクトコマンドオンラインモードでのオブジェクトハンドラ機能（アンロードやロードなど）の実行（「*バッチまたはダイレクトコマンド呼び出し*」も参照）。
- ワークプラン（「*ワークプラン*」セクションも参照）の実行または参照
- ワークプラン内の命令としての使用
- 画面間の移動
- 特殊機能の実行

このセクションでは、基本的なコマンド構文と個々の節、これらのタスクの実行に使用できるパラメータやオプションの設定について説明します。また、ダイレクトコマンドの使用例も示します。

このセクションの構文図で使用している記号については、『*システムコマンド*』ドキュメントの「*システムコマンド構文*」を参照してください。

このセクションでは、次のトピックについて説明します。

基本的なコマンド構文

select-clause

オブジェクトリスト **- LIST** ワークプラン

parameter-setting

option-setting

ダイレクトコマンドの使用例

ナビゲーションと特別な機能のためのコマンド

33 基本的なコマンド構文

このセクションでは、オブジェクトハンドラ機能およびタイプ PROCEDURE のワークプランを実行するために用意されているオブジェクトハンドラのダイレクトコマンドについて説明します。また、古いユーティリティ SYSTRANS からオブジェクトハンドラへの移行に使用するコマンドについても説明します。

このセクションの構文図で使用している変数値については、『オブジェクトハンドラ』ドキュメントの該当するセクションを参照してください。構文図で使用している記号については、『システムコマンド』ドキュメントの「システムコマンド構文」を参照してください。

```
EXECUTE (procedure-workplan)
```

タイプ PROCEDURE のワークプランを実行します。「ワークプラン」セクションも参照してください。

```
UNLOAD select-clause [parameter-setting] [option-setting]
```

parameter-setting に定義されたパラメータおよび *option-setting* に定義されたオプションを使用して、*select-clause* に定義されたオブジェクトをアンロードします。

```
LOAD select-clause [parameter-setting] [option-setting]
```

parameter-setting に定義されたパラメータおよび *option-setting* に定義されたオプションを使用して、*select-clause* に定義されたオブジェクトをロードします。

```
LOADALL [parameter-setting] [option-setting]
```

*parameter-setting*に定義されたパラメータおよび*option-setting*に定義されたオプションを使用して、ワークファイルからすべてのオブジェクトをロードします。

```
SCAN select-clause [option-setting]
```

*option-setting*に定義されたオプションを使用して、*select-clause*に定義されたオブジェクトのワークファイルをスキャンします。

```
SCANALL [option-setting]
```

*option-setting*に定義されたオプションを使用して、すべてのオブジェクトのワークファイルをスキャンします。

```
FIND select-clause [option-setting]
```

*option-setting*に定義されたオプションを使用して*select-clause*に定義されたオブジェクトを検索し、検出されたオブジェクトのレポートをワークファイル3に書き込みます。また、検出されたオブジェクトのレポートは、ワークファイル4、または指定されたレポートファイルに書き込むことができます。

```
FINDLIB select-clause [option-setting]
```

*option-setting*に定義されたオプションを使用して*select-clause*に定義されたNaturalオブジェクトまたはNaturalコマンドプロセッサソースのライブラリを検索し、検出されたオブジェクトのレポートをワークファイル3に書き込みます。また、検出されたオブジェクトのレポートは、ワークファイル4、または指定されたレポートファイルに書き込むことができます。

```
DELETE select-clause [option-setting]
```

*option-setting*に定義されたオプションを使用して、*select-clause*に定義されたオブジェクトを削除します。
制限事項FDTは削除できません。

```
UNDELI select-clause [option-setting]
```

*option-setting*に定義されたオプションを使用して、*select-clause*に定義されたオブジェクトの削除命令をアンロードします。

`RESTART [restart-file]`

中断されたロード機能を継続します。これは、中断されたロード中に情報が再スタートファイルに書き込まれた場合にのみ可能です。ロードの再スタート情報は、ワークファイル6または指定された再スタートファイルに書き込むことができます。*option-setting* セクションの RESTART（「*ダイレクトコマンド*」）および「*ロードの再スタート*」を参照してください。

`DISPLAY STATISTICS`

処理されたオブジェクトの統計情報が表示されます。

`NATUNLD natunld-direct-command`

古いユーティリティ NATUNLD の構文でオブジェクトハンドラコマンドを実行します。「***NATUNLD/NATLOAD** および **SYSTRANS** からオブジェクトハンドラへの移行*」も参照してください。

`NATLOAD natload-direct-command`

古いユーティリティ NATLOAD の構文で発行されたオブジェクトハンドラダイレクトコマンドを実行します。「***NATUNLD/NATLOAD** および **SYSTRANS** からオブジェクトハンドラへの移行*」も参照してください。

`SYSTRANS systrans-direct-command`

古いユーティリティ SYSTRANS の構文で発行されたオブジェクトハンドラダイレクトコマンドを実行します。「***NATUNLD/NATLOAD** および **SYSTRANS** からオブジェクトハンドラへの移行*」も参照してください。

34 select-clause

- select-clause の構文 194
- SELECTION または LIST ワークプラン 194
- Natural ライブラリオブジェクトと DDM の選択 195
- Natural 関連オブジェクトの選択 201
- Natural システムエラーメッセージの選択 203
- Natural コマンドプロセッサソースの選択 204
- 外部ファイルの選択 206
- FDT の選択 207
- アプリケーションの選択 208
- 削除命令のためのオブジェクト選択 211

select-clause は、タイプ SELECTION または LIST のワークプラン、あるいは処理するオブジェクト、FDT、またはアプリケーションの選択指定のいずれかで構成されます。

このセクションでは、*select-clause*に適用される構文について説明します。構文図で使用しているキーワードおよび変数値は、オブジェクトの選択条件の指定に使用できるパラメータを表します。必要な場合、変数値をキーワードと一緒に指定する必要があります。

select-clause の構文

select-clause は、次のいずれかのオプションで構成されます。

```
{ (selection-workplan)
  (list-workplan)
  object-selection
  delete-instruction-selection }
```

selection-workplan および *list-workplan* オプションについては、以下の「*SELECTION または LIST ワークプラン*」で説明しています。

*object-selection*の使用法は、処理するオブジェクトタイプ、DDM、FDT、またはアプリケーションによって異なります。それぞれに適した構文およびキーワードについて、このセクションの残りで説明します。

delete-instruction-selection オプションについては、「*削除命令のためのオブジェクト選択*」で説明しています。

SELECTION または LIST ワークプラン

タイプ SELECTION のワークプランには、ヘッダー（TYPE SELECTION）と、次のいずれかのタイプのオブジェクトまたはファイルからの選択内容が含まれます。Natural ライブラリオブジェクト、Natural 関連オブジェクト、Natural システムエラーメッセージ、Natural コマンドプロセッサソース、外部ファイル（外部オブジェクト）、または Adabas FDT（フィールド定義テーブル）。

タイプ LIST のワークプランには、ヘッダー（TYPE LIST）と「*オブジェクトリスト - LIST ワークプラン*」セクションで説明しているオブジェクトの選択リストが含まれます。このようなオブジェクトリストは、UNLOAD、LOAD、または FIND コマンドにのみ使用できます。

ワークプランの使用の詳細については、「*ワークプラン*」セクションを参照してください。

Natural ライブラリオブジェクトと DDM の選択

この選択は、Natural DDM、ユーザー定義エラーメッセージ、共有リソースなど、処理する Natural オブジェクトを選択するために使用します。

以下のセクションでは、適切な構文を示して説明します。

- Natural ライブラリオブジェクトと DDM の選択の構文

Natural ライブラリオブジェクトと DDM の選択の構文

```
object-name
LIBRARY library-name
[DBID dbid FNR fnr]
[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]
)
```

注意: コマンド FINDLIB では、次のキーワードのみが処理されます。LIBRARY, DBID および FNR.

Natural ライブラリオブジェクトと **DDM** の選択のキーワードの説明

次のセクションでは、処理するオブジェクトのキーワードおよび有効な値について説明します。

キーワード	有効な値	デフォルト値
object-name	有効なオブジェクト名または名前の範囲。 *object-name*に空白文字が含まれている場合、文字列を二重引用符（""）で囲む必要があります。 「*名前と日時の指定*」の「*名前*」も参照してください。	なし
LIBRARY	有効なライブラリ名または名前の範囲。 SETNO が指定されている場合、名前の範囲は使用できません。 「*名前*」も参照してください。	なし
DBID	有効なデータベース ID。	0

キーワード	有効な値	デフォルト値
		（現在の FNAT/FUSER）
FNR	有効なファイル番号。	0 （現在の FNAT/FUSER）
OBJTYPE	オブジェクトタイプは次のとおりです。 E ユーザー定義エラーメッセージ N Natural プログラミングオブジェクト R 共有リソース * アスタリスク（すべて） または有効な組み合わせ。	*
SETNO	Predictがインストールされている場合に、アンロード機能および検索機能にのみ適用されます。アプリケーションオブジェクトには適用されません（「*アプリケーションオブジェクトの選択*」も参照）。 処理するオブジェクトの名前に使用する保有セットを識別する 1 桁または 2 桁の数字。保有セットは、LIST XREF コマンドのセット保存オプションで作成されます。 SETNOが指定されていると、*object-name*に指定した値は無視されます。 Predict セットの詳細については、『*Predict*』ドキュメントを参照してください。	なし
SETUSER	Predictがインストールされている場合に、アンロード機能および検索機能にのみ適用されます。アプリケーションオブジェクトには適用されません（「*アプリケーションオブジェクトの選択*」も参照）。 Predict セットを作成したユーザーの ID。ID が指定されていないと、システム変数 *USER（『システム変数』ドキュメントも参照）の値が使用されます。	*USER
SETLIBRARY	Predictがインストールされている場合に、アンロード機能および検索機能にのみ適用されます。アプリケーションオブジェクトには適用されません（「*アプリケーションオブジェクトの選択*」も参照）。 Predict セットに使用するために検索するライブラリの名前。SETLIBRARY を指定しないと、LIBRARY で指定されたライブラリが代わりに使用されます。	
NATTYPE	Natural オブジェクトタイプの 1 つまたは複数の単一文字コード： P プログラム NSubprogram Sサブルーチン	*

キーワード	有効な値	デフォルト値
	Cコピーコード HHelproutine Tテキスト 7関数 8アダプタ Gグローバルデータエリア Lローカルデータエリア Aパラメータデータエリア Mマップ 4クラス 3 ダイアログ 5Natural コマンドプロセッサ VDDM *すべてのオブジェクトタイプ	
SCKIND	Natural プログラミングオブジェクトの種類。 有効な入力値は次のとおりです。 S ソースオブジェクト：ソース形式でのみ保存されるオブジェクト C カタログ化オブジェクト：カタログ形式でのみ保存されるオブジェクト A すべてのソースオブジェクトおよびカタログ化オブジェクト。 W STOW を実行された全オブジェクト：同じ日時のソースオブジェクトとカタログ化オブジェクト B ソースオブジェクトおよびカタログ化オブジェクト（両方ともある場合）。 **注意:** W と B は、UNLOAD および FIND コマンドにのみ有効です。LOAD および SCAN では、W と B は有効なエントリですが、A（すべてのオブジェクト）として処理されます。データが転送フォーマットで処理される場合、S（ソースオブジェクト）または A のみが適用されます。	A
MODE	Natural プログラミングオブジェクトのプログラミングモード。 有効な入力値は次のとおりです。 A 任意。 R レポーティングモードのすべてのオブジェクト。 S ストラクチャードモードのすべてのオブジェクト。	
FMNUM	Natural エラーメッセージの開始番号。 有効範囲：1～9999。	1

キーワード	有効な値	デフォルト値
TONUM	Natural エラーメッセージの終了番号。 有効範囲：1～9999。 値は、FMNUM の値（指定された場合）以上にする必要があります。	9999 または FMNUM （指定されている場合）
SLKIND	Natural エラーメッセージのテキストのタイプ。 有効な入力値は次のとおりです。 S 短文テキスト。 DELETE コマンドには適用できません（「*基本的なコマンド構文*」を参照）。 L 長文テキスト。 A 短文テキストと長文テキストの両方または片方。 B 短文テキストと長文テキスト（両方が存在する場合）。	A
LANGUAGE	ユーザー定義エラーメッセージの最大8つまでの有効な言語コード（英語の場合はコード1）。アスタリスク（*）にすると、すべての言語コードが選択されます。	*
DDMDBID	DDM の有効なデータベース ID（1～65535）。 UNLOAD、LOAD および SCAN：0 はチェックが実行されないことを表します。データベース ID（DBID）に関係なく、DDM は処理されます。	0
DDMFNR	DDM の有効なファイル番号（1～65535）。 UNLOAD、LOAD および SCAN：0 はチェックが実行されないことを表します。ファイル番号（FNR）に関係なく、DDM は処理されます。	0
NATVERS	Natural プログラミングオブジェクトの Natural バージョン。バージョンの範囲を指定することもできます。「*名前*」を参照してください。	空白 （チェックなし）
DATE	Natural プログラミングオブジェクトを保存またはカタログした日付、および共有リソースの日付。 時刻を追加するには、日付と時刻の間に空白を挿入します。有効なフォーマットと範囲については、「*名前と日時の指定*」の「*日付*」および「*時刻*」を参照してください。 日付には、YESTERDAY および TODAY という特殊な表現を使用できます。「*日付*」の「*特殊な日付の値*」を参照してください。	空白 （チェックなし）
FMDATE	開始値： Natural プログラミングオブジェクトがカタログまたは保存された日付、および共有リソースの日付は、この日付以降になります。フォーマットは DATE と同じです。「*日付*」を参照してください。 日付には、YEAR、MONTH、YESTERDAY および TODAY という特殊な表現を使用できます。「*日付*」の「*特殊な日付の値*」を参照してください。	空白 （チェックなし）

キーワード	有効な値	デフォルト値
TODATE	終了値： Natural プログラミングオブジェクトがカタログまたは保存された日付、および共有リソースの日付は、この日付以前になります。フォーマットは DATE と同じです。「*日付*」を参照してください。 日付には、YEAR、MONTH、YESTERDAY および TODAY という特殊な表現を使用できます。「*日付*」の「*特殊な日付の値*」を参照してください。	空白 （チェックなし） または上位の値 （FMDATE が指定されている場合）
SIZE	Natural プログラミングオブジェクトおよび共有リソースのサイズ（7 桁以内）。	0 （チェックなし）
FMSIZE	開始値： Natural プログラミングオブジェクトおよび共有リソースの最小サイズ（7 桁以内）。	0 （チェックなし）
TOSIZE	終了値： Natural プログラミングオブジェクトおよび共有リソースの最大サイズ（7 桁以内）。	0 （チェックなし） または上位の値 （FMSIZE が指定されている場合）
USERID	Natural プログラミングオブジェクトを保存またはカタログしたユーザーの ID。 ユーザー ID の範囲を指定することもできます。「*名前*」を参照してください。	空白 （チェックなし）
TID	Natural プログラミングオブジェクトが保存またはカタログされた端末の ID（Natural システム変数 *INIT-ID で指定される）。 端末 ID の範囲を指定することもできます。「*名前*」を参照してください。	空白 （チェックなし）
EXCEPT	EXCEPT の前に入力された選択条件に一致しているすべての項目は、キーワード EXCEPT の後のカッコ内に含まれているすべてのパラメータに対してチェックされます。すべてのパラメータとも一致している場合、処理されません。	該当なし

注意:

1. OBJTYPE に無関係なパラメータは無視されます。例えば、次のようになります。DATE、SIZE および USERID は、Natural エラーメッセージに対して意味がありません。
2. DBIDおよび FNR は、LOAD または SCAN コマンドでは無視されます。これらのパラメータは、代わりに *parameter-setting* 節で指定する必要があります（「*parameter-clause のキーワードの説明*」の「LOADFNAT...」および「LOADFUSER...」を参照）。
3. 共有リソースのオブジェクトに空白文字が含まれている場合、二重引用符（" "）で囲む必要があります。

Natural 関連オブジェクトの選択

この選択は、処理する Natural 関連オブジェクトを選択するために使用します。

以下のセクションでは、適切な構文を示して説明します。

- Natural 関連オブジェクトの選択の構文

Natural 関連オブジェクトの選択の構文

```
object-name NATPATH natural-path-name
[      DATE date                                                  ]
       [FMDATE date-from] [TODATE date-to]
[      SIZE size                                                  ]
       [FMSIZE size-from] [TOSIZE size-to]
[ EXCEPT
       (object-name NATPATH natural-path-name
       [                                DATE date                 ]
                                        [FMDATE date-from] [TODATE date-to]
       [                                SIZE size                 ]
                                        [FMSIZE size-from] [TOSIZE size-to]
       )]
```

Natural 関連オブジェクトの選択のキーワードの説明

次のセクションでは、処理するオブジェクトのキーワードおよび有効な入力値について説明します。

キーワード	有効な値	デフォルト値
object-name	Natural関連オブジェクトの名前。*object-name*に空白文字が含まれている場合、文字列を二重引用符（""）で囲む必要があります。 「*名前と日時の指定*」の「*名前*」も参照してください。	なし
NATPATH	NATROOT NATGUI_BMP TMP_PATH NATBIN PROFILE_PATH PARM_PATH NATERR	なし

キーワード	有効な値	デフォルト値
DATE	Natural 関連オブジェクトの修正日付。 時刻を追加するには、日付と時刻の間に空白を挿入します。有効なフォーマットと範囲については、「*名前と日時の指定*」の「*日付*」および「*時刻*」を参照してください。 日付には、次の特殊な表現を使用できます。YESTERDAY および TODAY 「*日付*」の「*特殊な日付の値*」を参照してください。	空白 （チェックなし）
FMDATE	開始値： Natural 関連オブジェクトが修正された日付は、この日付以降になります。フォーマットは DATE と同じです。「*日付*」を参照してください。 日付には、次の特殊な表現を使用できます。YEAR、MONTH、YESTERDAY、および TODAY「*日付*」の「*特殊な日付の値*」を参照してください。	空白 （チェックなし）
TODATE	終了値： Natural 関連オブジェクトが修正された日付は、この日付以前になります。フォーマットは DATE と同じです。「*日付*」を参照してください。 日付には、次の特殊な表現を使用できます。YEAR、MONTH、YESTERDAY、および TODAY「*日付*」の「*特殊な日付の値*」を参照してください。	空白 （チェックなし） または上位の値 （FMDATE が指定されている場合）
SIZE	Natural 関連オブジェクトのサイズ（10 桁以内）。	0 （チェックなし）
FMSIZE	開始値： Natural 関連オブジェクトの最小サイズ（10 桁以内）。	0 （チェックなし）
TOSIZE	終了値： Natural 関連オブジェクトの最大サイズ（10 桁以内）。	0 （チェックなし） または上位の値 （FMSIZE が指定されている場合）
EXCEPT	「*Natural ライブラリオブジェクトと DDM の選択*」の「EXCEPT」を参照してください。	

注意: EXCEPT 部分の NATPATH 節は、LOAD または SCAN コマンドでのみ評価されます。

Natural システムエラーメッセージの選択

この選択は、処理する Natural システムエラーメッセージを選択するために使用します。

以下のセクションでは、適切な構文を示して説明します。

- Natural システムエラーメッセージの選択の構文

Natural システムエラーメッセージの選択の構文

```
ERROR NATERROR
[DBID dbid FNR fnr]
[FMNUMerror-number-from] [TONUMerror-number-to]
[SLKINDmessage-type]
[LANGUAGElanguages]
[ EXCEPT
                                                    (
                                                    [FMNUMerror-number-from]
                                                    [TONUMerror-number-to]
                                                    [SLKINDmessage-type]
                                                    [LANGUAGElanguages]
                                                    ) ]
```

Natural システムエラーメッセージの選択のキーワードの説明

次のセクションでは、処理する Natural システムエラーメッセージのキーワードおよび有効な入力値について説明します。

キーワード	有効な値	デフォルト値
DBID	メインフレーム上のシステムエラーメッセージにのみ適用されます。 有効なデータベース ID。	0 （現在の FNAT）
FNR	メインフレーム上のシステムエラーメッセージにのみ適用されます。 有効なファイル番号。	0 （現在の FNAT）
FMNUM	システムエラーメッセージの開始番号。 有効範囲：1～9999。	1

キーワード	有効な値	デフォルト値
TONUM	システムエラーメッセージの終了番号。 有効範囲：1～9999。 値は、FMNUMの値（指定された場合）以上にする必要があります。	9999 またはFMNUMの値 （指定されている場合）
SLKIND	「*Natural ライブラリオブジェクトとDDMの選択*」の「SLKIND」を参照してください。	A
LANGUAGE	システムエラーメッセージの最大8つまでの有効な言語コード（英語の場合はコード1）。 アスタリスク（*）にすると、すべての言語コードが選択されます。	*
EXCEPT	「*Natural ライブラリオブジェクトとDDMの選択*」の「EXCEPT」を参照してください。	

注意: DBIDおよびFNRは、LOADまたはSCANコマンドでは無視されます。これらのパラメータは、代わりに*parameter-setting*節で指定する必要があります（「*parameter-clauseのキーワードの説明*」の「LOADFNAT...」を参照）。

Naturalコマンドプロセッサソースの選択

この選択は、処理するNaturalコマンドプロセッサソースを選択するために使用します。

以下のセクションでは、適切な構文を示して説明します。

- Naturalコマンドプロセッサソースの選択の構文

Naturalコマンドプロセッサソースの選択の構文

```
object-name PROCESSORncp-library-name
[            DBID ncp-dbid FNR ncp-fnr [file-options]]
[EXCEPT
             (object-name
             [LIBRARYncp-library-name]
             )]
```

file-options

```
[CIPHERncp-cipher]
[ { PASSWORD } ncp-password ]
  { PSW      }
```

注意: コマンド FINDLIB では、次のキーワードのみが処理されます。PROCESSOR、DBID、FNR、CIPHER、および PASSWORD または PSW

Natural コマンドプロセッサソースの選択のキーワードの説明

次のセクションでは、処理する Natural コマンドプロセッサソースのキーワードおよび有効な入力値について説明します。

キーワード	有効な値	デフォルト値
object-name	Natural コマンドプロセッサソース名または名前の範囲。 「*名前と日時の指定*」の「*名前*」も参照してください。	なし
PROCESSOR	有効なライブラリ名または名前の範囲。 「*名前*」も参照してください。	なし
DBID	Natural コマンドプロセッサソースが保存されている Adabas ファイルの有効なデータベース ID。	LFILE 190 の値
FNR	Natural コマンドプロセッサソースが保存されている Adabas ファイルの有効なファイル番号。	LFILE 190 の値
CIPHER	Natural コマンドプロセッサソースが保存されている Adabas ファイルの 8 桁のサイファコード。	空白
PASSWORD または PSW	Natural コマンドプロセッサソースが保存されている Adabas ファイルの 8 文字の Adabas パスワード。	空白
EXCEPT	「*Natural ライブラリオブジェクトと DDM の選択*」の「EXCEPT」を参照してください。	

注意: DBID、FNR、、CIPHER、および PASSWORD または PSW は、LOAD または SCAN コマンドでは無視されます。これらのパラメータは、代わりに *parameter-setting* 節で指定する必要があります（「*parameter-clause のキーワードの説明*」の「LOADNCP...」を参照）。

外部ファイルの選択

この選択は、処理する外部ファイル（外部オブジェクト）を選択するために使用します。

以下のセクションでは、適切な構文を示して説明します。

- 外部ファイルの選択の構文

外部ファイルの選択の構文

```
external-file-name PATHexternal-path-name
[       DATE date                                                   ]
[       [FMDATE date-from] [TODATEdate-to]                          ]
[       SIZE size                                                   ]
[       [FMSIZE size-from] [TOSIZEsize-to]                          ]
[ EXCEPT
        (external-file-name [PATHexternal-path-name]
        [                                   DATE date               ]
        [                                   [FMDATE date-from] [TODATEdate-to] ]
        [                                   SIZE size               ]
        [                                   [FMSIZE size-from] [TOSIZEsize-to] ]
        )]
```

外部ファイルの選択のキーワードの説明

次のセクションでは、処理する外部ファイルのキーワードおよび有効な入力値について説明します。

キーワード	有効な値	デフォルト値
external-file-name	外部ファイルの名前。 *external-file-name*に空白文字が含まれている場合、文字列を二重引用符（" "）で囲む必要があります。 「*名前と日時の指定*」の「*名前*」も参照してください。	なし
PATH	外部ファイルが保存されているパスの名前。	なし
DATE	外部ファイルの修正日付。 時刻を追加するには、日付と時刻の間に空白を挿入します。有効なフォーマットと範囲については、「*名前と日時の指定*」の「*日付*」および「*時刻*」を参照してください。	空白 （チェックなし）

キーワード	有効な値	デフォルト値
	日付には、YESTERDAY および TODAY という特殊な表現を使用できます。「*日付*」の「*特殊な日付の値*」を参照してください。	
FMDATE	開始値： 外部ファイルが修正された日付は、この日付以降になります。フォーマットは DATE と同じです。「*日付*」を参照してください。 日付には、YEAR、MONTH、YESTERDAY および TODAY という特殊な表現を使用できます。「*特殊な日付の値*」を参照してください。	空白 （チェックなし）
TODATE	終了値： 外部ファイルが修正された日付は、この日付以前になります。フォーマットは DATE と同じです。「*日付*」を参照してください。 日付には、YEAR、MONTH、YESTERDAY および TODAY という特殊な表現を使用できます。「*特殊な日付の値*」を参照してください。	空白 （チェックなし） または上位の値 （FMDATE が指定されている場合）
SIZE	外部ファイルのサイズ（10 桁以内）。	0 （チェックなし）
FMSIZE	開始値： 外部ファイルの最小サイズ（10 桁以内）。	0 （チェックなし）
TOSIZE	終了値： 外部ファイルの最大サイズ（10 桁以内）。	0 （チェックなし） または上位の値 （FMSIZE が指定されている場合）
EXCEPT	「*Natural ライブラリオブジェクトと DDM の選択*」の「EXCEPT」を参照してください。	

注意: EXCEPT 部分の NATPATH 節は、LOAD および SCAN コマンドでのみ評価されます。

FDT の選択

この選択は、処理する Adabas FDT（フィールド定義テーブル）を選択するために使用します。

FDT のロードの詳細については、「*オブジェクト指定*」セクションの「***FDT***」も参照してください。

以下のセクションでは、適切な構文を示して説明します。

- FDT の選択の構文

FDT の選択の構文

```
FDT
DBID dbid
{ FNR fnr [CIPHER cipher] [ { PASSWORD } password ] }
{                         { PSW      }            }
{ FMFNR fnr-start TOFNR fnr-end                     }
```

FDT の選択のキーワードの説明

次のセクションでは、処理する FDT のキーワードおよび有効な入力値について説明します。

キーワード	有効な値	デフォルト値
DBID	FDT のデータベース ID。	なし
FNR	FDT のファイル番号。	なし
CIPHER	FDTの 8 桁の Adabas サイファコード。	なし
PASSWORD または PSW	FDT の 8 文字の Adabas パスワード。	なし
FMFNR	FIND または UNLOAD コマンドにのみ適用されます。 開始値： FDT のファイル番号（FNR）。	なし
TOFNR	FIND または UNLOAD コマンドにのみ適用されます。 終了値： FDT のファイル番号（FNR）。	なし

アプリケーションの選択

この選択は、Naturalスタジオのアプリケーションワークスペースで作成および管理されるアプリケーション、およびそのアプリケーションに属するライブラリまたはオブジェクトに適用されます。

以下のセクションでは、適切な構文を示して説明します。

- 基本アプリケーションと複合アプリケーションの選択
- アプリケーションライブラリの選択

- アプリケーションオブジェクトの選択

基本アプリケーションと複合アプリケーションの選択

この選択内容は検索機能にのみ適用されます。

構文

```
APPLICATION APNAMEapplication-name
[APTYPEapplication-type]
[COMPAPPLICATIONcompound-application-name]
[                    EXCEPT
                     (APNAMEapplication-name
                     [APTYPEapplication-type]
                     )]
```

アプリケーションライブラリの選択

この選択内容は検索機能にのみ適用されます。

構文

```
APPLICATION  APLIBRARYapplication-library-name
[BASEAPPLICATIONbase-application-name]
[COMPAPPLICATIONcompound-application-name]
[            DBID dbid [FNR fnr] ]
[            EXCEPT
             (APLIBRARYapplication-library-name
             [BASEAPPLICATIONbase-application-name]
             )]
```

アプリケーションオブジェクトの選択

このセクションは、検索およびアンロード機能にのみ適用されます。

構文

```
APPLICATION  APOBJECTSapplication-object-name
[BASEAPPLICATIONbase-application-name]
[COMPAPPLICATIONcompound-application-name]
[LIBRARYlibrary-name]
[object-specification]
[                        EXCEPT
                         (APOBJECTapplication-object-name
                         [LIBRARYlibrary-name]
                         [BASEAPPLICATIONbase-application-name]
                         [object-specification]
                         )]
```

アプリケーションの選択のキーワードの説明

次のセクションでは、処理するアプリケーション、アプリケーションライブラリ、またはアプリケーションオブジェクトのキーワードおよび有効な入力値について説明します。

キーワード	有効な値	デフォルト値
APNAME	Natural アプリケーションの有効な名前または名前の範囲。 「*名前と日時の指定*」の「*名前*」も参照してください。	*
APTYPE	有効なアプリケーションタイプ： B：基本アプリケーション O：複合アプリケーション *：All：基本アプリケーションまたは複合アプリケーションあるいはその両方	*
COMPAPPLICATION	APTYPE が * または B に設定されている場合にのみ適用されます。 指定された基本アプリケーションが属す複合アプリケーションの名前または名前の範囲。 指定された複合アプリケーションに属す基本アプリケーションのみが選択され、複合アプリケーションに属さない基本アプリケーションは選択されません。	なし
EXCEPT	「*Natural ライブラリオブジェクトと DDM の選択*」の「EXCEPT」を参照してください。	該当なし

キーワード	有効な値	デフォルト値
APLIBRARY	Natural基本アプリケーションまたは複合アプリケーションに属すライブラリの有効な名前または名前の範囲。 「*名前と日時の指定*」の「*名前*」も参照してください。	*
BASEAPPLICATION	アプリケーションライブラリまたはアプリケーションオブジェクトが属す Natural 基本アプリケーションの有効な名前。 「*名前と日時の指定*」の「*名前*」も参照してください。	*
DBID	アプリケーションライブラリの有効なデータベース ID。	0 （チェックなし）
FNR	アプリケーションライブラリの有効なファイル番号。	0 （チェックなし）
APOBJECT	基本アプリケーションまたは複合アプリケーションに属すアプリケーションオブジェクトの有効な名前または名前の範囲。 「*名前と日時の指定*」の「*名前*」も参照してください。	*
LIBRARY	有効なライブラリ名または名前の範囲。 OBJTYPE が D（「*Natural ライブラリオブジェクトと DDM の選択*」を参照）に設定されている場合、ライブラリ名は無視されます。 「*名前と日時の指定*」の「*名前*」も参照してください。	*
object-specification	Natural ライブラリオブジェクトと DDM の構文図に示すように、アプリケーションオブジェクトに追加の選択条件を指定できることを示します。LIBRARY LIBRARY *library-name* の下に示しているすべての項目はアプリケーションオブジェクトにも適用できますが、EXCEPT 節の *object-name* はアプリケーションオブジェクトには無関係です。	該当なし

削除命令のためのオブジェクト選択

この選択は、Natural ライブラリオブジェクト、DDM、ユーザー定義エラーメッセージ、および Natural システムエラーメッセージに対する削除命令を指定する場合に使用します。この削除命令は、DELETEALLOWED オプションを指定して、内部フォーマットのワークファイルがターゲット環境にロードされると実行されます。

以下のセクションでは、適切な構文を示して説明します。

- Natural ライブラリオブジェクトと DDM の削除命令の構文
- ユーザー定義エラーメッセージの削除命令の構文

- Natural システムエラーメッセージの削除命令の構文

Natural ライブラリオブジェクトと DDM の削除命令の構文

```
object-name
LIBRARY library-name
[ OBJTYPE { N } ]
[ NATTYPE { * } ]
          { V }
[SCKIND object-kind]
```

Natural ライブラリオブジェクトと **DDM** の削除命令のキーワードの説明

次のセクションでは、処理するオブジェクトのキーワードおよび有効な値について説明します。

キーワード	有効な値	デフォルト値
object-name	有効なオブジェクト名、または名前範囲の開始値（*value**）（例えば、ABC*）	なし
LIBRARY	有効なライブラリ名。範囲の指定はできません。	なし
OBJTYPE	有効なオブジェクトタイプコード： N Natural プログラミングオブジェクト	*
NATTYPE	Natural オブジェクトタイプ。有効な入力値は次のとおりです。 *すべてのオブジェクトタイプ VDDM	*
SCKIND	Natural プログラミングオブジェクトの種類。 有効な入力値は次のとおりです。 S ソースオブジェクト。*except-clause* で使用した場合（「*Natural ライブラリオブジェクトと DDM の選択の構文*」を参照）：ソース形式のみで保存されるオブジェクト。 C カタログ化オブジェクト。*except-clause* で使用した場合、カタログ形式のみで保存されるオブジェクト。 A すべてのソースオブジェクトおよびカタログ化オブジェクト。	A

ユーザー定義エラーメッセージの削除命令の構文

```
*
LIBRARY library-name
OBJTYPE E
FMNUM error-number-from
[TONUM error-number-to]
[SLKIND message-type]
[LANGUAGE languages]
```

library-name は、単一のライブラリの名前です。範囲の指定はできません。

この構文で使用される他の要素については、「*Natural ライブラリオブジェクトと DDM の選択のキーワードの説明*」を参照してください。

Natural システムエラーメッセージの削除命令の構文

```
ERROR NATERROR
FMNUM error-number-from
[TONUM error-number-to]
[SLKIND message-type]
[LANGUAGE languages]
```

この構文で使用される要素については、「*Natural システムエラーメッセージの選択のキーワードの説明*」を参照してください。

35 オブジェクトリスト - LIST ワークプラン

- object-type-and-location の構文 216
- object-name-description の構文 217
- オブジェクトリストの例 218

オブジェクトリストとはタイプ LIST のワークプランのことであり、UNLOAD、LOAD、FIND または DELETE コマンドで処理するオブジェクトに対してオブジェクトの選択条件を指定します。オブジェクトリストは、*select-clause* および SELECTION ワークプランの代わりに使用できます。

オブジェクトリストに適用される構文は次のとおりです。

```
TYPE LIST
{ object-type-and-location (object-name-description ...) } ...
```

構文オプションについては、次のセクションで説明します。このセクションの構文図で使用しているキーワードおよび変数値は、オブジェクトの選択条件の指定に使用されるパラメータを表します。必要な場合、変数値をキーワードと一緒に指定する必要があります。各構文要素（カッコ内の構文要素を除く）は、新しい行で開始し、同じ行で終了する必要があります。

構文図で使用しているキーワードについては、「*select-clause*」セクションを参照してください。

object-type-and-location の構文

次のセクションでは、*object-type-and-location* に適用される構文図を示します。

- Natural オブジェクトと DDM
- Natural システムエラーメッセージ
- Natural コマンドプロセッサソース
- Natural 関連オブジェクト
- 外部ファイル（外部オブジェクト）
- FDT

Natural オブジェクトと DDM

```
LIBRARY library-name
  [DBID dbid FNR fnr]
[OBJTYPE group-type]
```

注意: *library-name* には範囲は使用できません。

Natural システムエラーメッセージ

```
ERROR NATERROR
[DBID dbid FNR fnr]
```

Natural コマンドプロセッサソース

```
PROCESSORncp-library-name
  [ DBID dbid FNR fnr [CIPHERcipher] [ { PASSWORD } password ] ]
                                       { PSW      }
```

注意: *ncp-library-name* には範囲は使用できません。

Natural 関連オブジェクト

```
NATPATHnatural-path-name
```

外部ファイル（外部オブジェクト）

```
PATHexternal-path-name
```

FDT

```
FDT
```

object-name-description の構文

次のセクションでは、*object-name-description* に適用される構文図を示します。

- Natural オブジェクト
- Natural システムエラーメッセージ
- Natural コマンドプロセッサソース
- Natural 関連オブジェクト
- 外部ファイル（外部オブジェクト）

- FDT

Natural オブジェクト

```
{ 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] }
```

Natural システムエラーメッセージ

```
{ error-number [SLKIND message-type] [LANGUAGE languages]
  FMNUM error-number-from TONUM error-number-to [SLKIND message-type] [LANGUAGE languages] }
```

Natural コマンドプロセッサソース

```
object-name
```

Natural 関連オブジェクト

```
related-object-name
```

外部ファイル（外部オブジェクト）

```
external-file-name
```

FDT

```
DBID dbid FNR fnr [CIPHER cipher] [ { PASSWORD } password ]
                                    { PSW      }
```

オブジェクトリストの例

次に、タイプ LIST のワークプランの例を示します。

```
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
  )
```

36 parameter-setting

- parameter-clause の構文 222
- parameter-clause のキーワードの説明 223

parameter-setting 節は、オブジェクトを処理する LOAD または UNLOAD コマンドの属性を変更し、LOAD コマンドの出力宛先（FNAT など）を定義するために使用します。

parameter-setting 節には次の構文が適用されます。

```
WITH
{  (parameter-workplan)  }
{  parameter-clause      }
```

parameter-workplan に適用される構文については、「ワークプラン」セクションの「ワークプランの参照」を参照してください。

このセクションでは、次のトピックについて説明します。

parameter-clause の構文

以下の図は、*parameter-clause* の構文を示します。必要な場合、変数値をキーワードと一緒に指定する必要があります。

```
[        [NAMEold-name] NEWNAMEnew-name                                  ]
[        [LIBRARYold-library-name]                                       ]
         NEWLIBRARYnew-library-name
[        LOADFNATDBIDfnat-dbid LOADFNATFNRfnat-fnr                       ]
[        LOADFUSERDBIDfuser-dbid LOADFUSERFNR fuser-fnr                  ]
[        LOADNCPDBIDncp-file-dbid LOADNCPFNR ncp-file-fnr                ]
         [LOADNCPCIPHERncp-file-cipher]
    [    {              LOADNCPPASSWORD}     ncp-file-password           ]
         {              LOADNCPPSW      }
[        [FDTDBIDold-fdt-dbid FDTFNR old-fdt-fnr] NEWFDTDBIDnew-fdt-dbid ]
         NEWFDTFNRnew-fdt-fnr
[ERRNUMDIFFmodification-of-error-message-range]
[        [LANGUAGEold-language]                                          ]
         NEWLANGUAGEnew-language
[        [DATEold-date] NEWDATEnew-date                                  ]
[        [USERIDold-userid]                                              ]
         NEWUSERIDnew-userid
```

```
[        [TIDold-terminal-id]          ]
         NEWTIDnew-terminal-id
[        [PATHold-external-path-name]  ]
         NEWPATHnew-external-path-name
```

parameter-clause のキーワードの説明

次のセクションでは、*parameter-clause* のキーワードおよび変数値（該当する場合）について説明します。

キーワード	値	利用可能なコマンド
NAME	NEWNAME が指定されている場合にチェックされるオブジェクト名。	
NEWNAME	新しいオブジェクト名。 **注意:** メインフレームプラットフォーム上の DDM には適用されません。	
LIBRARY	NEWLIBRARY が指定されている場合にチェックされるライブラリ名。	
NEWLIBRARY	新しいライブラリ名。 **LOAD** 機能に関する注意事項： NEWLIBRARYは、DELETEALLOWED オプションを使用して処理されたワークファイルの削除命令で使用されるライブラリ名には*影響を及ぼしません*。	
LOADFNATDBID	FNAT ライブラリのデータベース ID（DBID）。	LOAD
LOADFNATFNR	FNAT ライブラリのファイル番号（FNR）。	LOAD
LOADFUSERDBID	FUSER ライブラリの DBID。	LOAD
LOADFUSERFNR	FUSER ライブラリの FNR。	LOAD
LOADNCPDBID	Natural コマンドプロセッサソース用 Adabas ファイルの DBID。	LOAD
LOADNCPFNR	Natural コマンドプロセッサソース用 Adabas ファイルの FNR。	LOAD
LOADNCPCIPHER	Natural コマンドプロセッサソース用 Adabas ファイルのサイファコード。	LOAD
LOADNCPPASSWORD または LOADNCPPSW	メインフレーム上のオブジェクトにのみ適用されます。 Natural コマンドプロセッサソース用 Adabas ファイルの Adabas パスワード。	LOAD
FDTDBID	NEWFDTDBIDが指定されている場合にチェックされる Adabas FDT（フィールド定義テーブル）の DBID。	
NEWFDTDBID	FDT の新しい DBID。	
FDTFNR	NEWFDTFNR が指定されている場合にチェックされる FDT の DBID。	
NEWFDTFNR	FDT の新しい FNR。	

キーワード	値	利用可能なコマンド
ERRNUMDIFF	UNLOAD または LOAD コマンドの実行中に Natural エラーメッセージに追加される数字（正または負）。 ERRNUMDIFF は、選択条件として FMNUM および TONUM（「*select-clause*」を参照）が指定された場合にのみ指定できます。指定されていない場合は、有効な結果をチェックできません。	
LANGUAGE	NEWLANGUAGE（下記参照）が指定されている場合にチェックされる Natural エラーメッセージの最大8つまでの有効な言語コード（英語の場合はコード 1）。 *language*に複数の言語コードが含まれている場合、*new-language*には同じ数の言語コードが含まれている必要があります。各 *language* 言語コードは、対応する位置の *new-language* で置き換えられます。 *language*が指定されていない場合、*new-language*に複数の言語コードを含めることはできません。	
NEWLANGUAGE	新しいユーザー定義エラーメッセージの最大8つまでの有効な言語コード（スペイン語の場合はコード4）。このオプションは、英語（言語コード 1）が唯一の有効な言語である長文テキストの Natural システムエラーメッセージには適用されません。 上記の「LANGUAGE」も参照してください。	
DATE	オブジェクト日付。 時刻を追加するには、日付と時刻の間に空白を挿入します。有効なフォーマットと範囲については、「*名前と日時の指定*」の「*日付*」および「*時刻*」を参照してください。	
NEWDATE	新しいオブジェクト日付。 NEWDATE には、日付の後に時刻を続けることができます。時刻を追加するには、日付と時刻の間に空白を挿入します。「*名前と日時の指定*」の「*日付*」および「*時刻*」も参照してください。	
USERID	NEWUSERID が指定されている場合にチェックされるユーザー ID。	
NEWUSERID	新しいユーザー ID。	
TID	メインフレーム上のオブジェクトにのみ適用されます。 NEWTID が指定されている場合にチェックされる端末 ID。	
NEWTID	メインフレーム上のオブジェクトにのみ適用されます。 新しい端末 ID。	
PATH	NEWPATH が指定されている場合にチェックされるパス名。	
NEWPATH	新しいパス名。	

注意:

1. 処理される選択条件に適用できないパラメータは無視されます。
2. LOADFNAT...、LOADFUSER...、および LOADNCP... は、LOAD コマンドにのみ使用され、他のコマンドでは無視されます。
3. LOADFNAT...は、SYS で始まるライブラリ（SYSTEM を除く）に対して使用されます。
4. LOADFUSER... は、SYS で始まるライブラリを除くライブラリ（SYSTEM は含まれる）に対して使用されます。
5. LOADNCP... は、Natural コマンドプロセッサソースに対して使用されます。

37 option-setting

- option-setting の構文 228
- option-setting のキーワードの説明 230

option-setting 節は、オブジェクトハンドラコマンドオプションのデフォルト値を変更するために使用します。

以下のセクションでは、*option-setting* 節に適用される構文を示して説明します。構文図で使用しているキーワードおよび変数値は、デフォルト値の指定に使用されるパラメータを表します。必要な場合、変数値をキーワードと一緒に指定する必要があります。

option-setting の構文

```
WHERE
{   (option-workplan)  }
{
{  option-clause       }
```

*option-workplan*に適用される構文図については、「ワークプラン」セクションの「*ワークプランの参照*」を参照してください。

以下のセクションでは、*option-clause* の構文を示します。

- **option-clause の構文**

option-clause の構文

```
[ REPLACE {      ALL      }      ]
          {      OBSOLETE }
          {      EXCEPT   }

[ transfer-options                ]
  internal-format-options

[ NOREPORT                        ]
  NEWREPORT [file-name]
  REPORT [file-name]
  BATCHREPORT

[ REPORT-OPTION-1 {      A }      ]
                  {      E }
                  {      S }

[ REPORT-FORMAT   {      S }      ]
                  {      T }

[ REPORT-MODE     {      S }      ]
                  {      L }
```

```
[ NORESTART                          ]
[ RESTART [restart-file]             ]
[NUMBERPROCESSnumber]
[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               }                     ]
[ USE-FDDM { YES } ]
[          { NO  } ]
[ { NEWWORKFILE } file-name [ { WORKFILETYPE } { DEFAULT  } ] ]
[ { WORKFILE    }           [ { WFTYPE       } { PORTABLE } ] ]
                                              { UTF-8   }
[ADAFDUWORKFILEfile-name]
```

セパレータ

FDIC および FSEC キーワードに続く各値の間に、または値がない場合に、セパレータとしてコンマを使用する必要があります。例えば、次のようになります。FDIC (10,21,,2a)。

セッションパラメータ ID（『パラメータリファレンス』ドキュメントの「*ID - 入力デリミタ文字*」を参照）がコンマに設定されている場合、値を区切るセパレータとしてスラッシュ（/）を使用します。

transfer-options

```
TRANSFER
[ CONVERSION-TABLE { SYSTEM-TABLE } ]
                   { USER-TABLE   }
                   [conversion-program]
[SUBSTITUTE]
[INCLUDE-LINE-NUMBERS]
[UPPERCASE-TRANSLATION]
[INCORPORATE-FREE-RULES]
[LOAD-CODE-PAGEcode-page-name]
[DA-FORMATdata-area-format]
```

internal-format-options

```
[ XREF { ON     } ]
       { OFF    }
       { DOC    }
       { FORCE  }
       { SPECIAL } ]
[DELETEALLOWED]
```

option-setting のキーワードの説明

次のセクションでは、*option-setting* のキーワードおよび変数値（該当する場合）について説明します。

オプション	説明	利用可能なコマンド
REPLACE	指定されたオプションに従って、既存のオブジェクトを置き換えます。 ALL　すべてのオブジェクト（デフォルト設定）。 OBSOLETE　ロードファイル内のオブジェクトの日付より古い日付になっているすべてのオブジェクト。 EXCEPT　ロードファイル内のオブジェクトの日付より新しい日付になっているオブジェクトを除いた、すべてのオブジェクト。	LOAD LOADALL
TRANSFER	転送モードを設定します。 データは、転送フォーマットで読み込みおよび書き込みが行われます。有効なオプションについては、「*transfer-options のキーワードの説明*」を参照してください。	UNLOAD LOAD SCAN
NOREPORT	以下のレポートファイル設定を指定します。 データはレポートファイルに記録されません。これは、FIND および FINDLIB コマンドに対するデフォルト設定です。	
NEWREPORT	以下のレポートファイル設定を指定します。 レポートデータは記録され、ワークファイル4または *file-name*に書き込まれます既存のファイルは上書きされます。	

オプション	説明	利用可能なコマンド
REPORT	以下のレポートファイル設定を指定します。 レポートデータは記録され、ワークファイル4または *file-name*に書き込まれますこれは、UNLOAD、LOAD、LOADALL、SCAN、SCANALL および DELETE コマンドに対するデフォルト設定です。	
BATCHREPORT	バッチ処理時または OBJHAPI アプリケーションプログラミングインターフェイスの使用時のレポート設定を指定します。 レポートデータは、それぞれSYSOUTに書き込まれるか、または画面に出力されます（レポートデータはファイルに書き込まれません）。	
REPORT-OPTION-1 または REPOPT1 または REP-OPT-1	ダイレクトコマンドが実行され、レポートを書き込むときに使用するレポートオプションを指定します。 Aすべてのレポート項目を表示します（デフォルト）。 Eエラーメッセージのみを表示します。これには、Natural Security からのメッセージ、および LOAD コマンドの実行中に発生したメッセージ（例：「「置換されませんでした」」）が含まれます。 Sバッチモードの場合のみ。エラーレポートは、次の 2 つの部分に分割されています。エラーメッセージを除くレポート項目は、デフォルトのレポートデバイス（REPORT(0)/CMPRINT）に書き込まれ、エラーメッセージ（Natural Security からのメッセージおよび LOAD コマンドの実行中に発生したメッセージを含む）は、2 番目のレポートデバイス（REPORT(1)/CMPRT01）に書き込まれます。オンラインモードでは、S は A と同じ効果があることに注意してください。	UNLOAD LOAD SCAN DELETE
REPORT-FORMAT または REPFRMT	ダイレクトコマンドが実行され、レポートを書き込むときに使用するレポートフォーマットを指定します。 Sアンロードレポートのソースデータ（ライブラリ名などのパラメータが変更される前にアンロードされたオブジェクトのデータ）を表示します（デフォルト設定）。 Tアンロードレポートのターゲットデータ（ライブラリ名などのパラメータが変更された後のデータ）を表示します。	UNLOAD
REPORT-MODE または REPM	ダイレクトコマンドが実行され、レポートを書き込むときに使用するレポートモードを指定します。 Sショートレポートを作成します。最も関連性の高いデータが、レポート行の最初の80列に短いデリミタとともに表示されます。 Lラージレポートを作成します。データは元の順序で大きいデリミタ（デフォルト設定）で表示されます。	
NORESTART	再スタート情報はファイルに書き込まれません。	LOAD

オプション	説明	利用可能なコマンド
RESTART	再スタート情報は、ワークファイル 6 または *restart-file*に書き込まれます。	LOAD
NUMBERPROCESS	処理するオブジェクトの数を指定します。 指定された数を処理した後、LOAD または SCAN コマンドは停止します。	LOAD SCAN
FIXEDLENGTH	アンロードワークファイルのフォーマットを固定サイズの最大レコード長に設定します。 内部フォーマットで書き込まれる場合、各データレコードには 256 バイトが含まれ、転送フォーマットでは 100 バイトが含まれます。	UNLOAD
FDIC	処理に使用するシステムファイル FDIC を指定します。 Adabas ファイルのデータベース ID（*dbid*）、ファイル番号（*fnr*）、パスワード（*password*）、およびサイファコード（*cipher*）。 値が指定されていない場合、または「0」が指定されている場合、現在の FDIC システムファイルが使用されます。	UNLOAD LOAD DELETE
FSEC	処理に使用するシステムファイル FSEC を指定します。 Adabas ファイルのデータベース ID（*dbid*）、ファイル番号（*fnr*）、パスワード（*password*）、およびサイファコード（*cipher*）。 値が指定されていない場合、または「0」が指定されている場合、現在の FSEC システムファイルが使用されます。	UNLOAD LOAD DELETE
USE-FDDM	FDDM システムファイルを処理に使用することを指定します。以下の「***USE-FDDM** のキーワードの説明*」を参照してください。	UNLOAD LOAD FIND DELETE
NEWWORKFILE または WORKFILE	使用するワークファイルを指定します。 UNLOAD または LOAD データが Natural ワークファイル 1 との間で転送されます。NEWWORKFILE が指定されると、既存のワークファイルの内容がデータで上書きされるか、または新しいワークファイルが先頭から埋められます。これに該当しない場合、データが追加されます。	UNLOAD LOAD SCAN
WORKFILETYPE または WFTYPE	LOAD および SCAN コマンドを使用する場合には設定する必要はありません。これらのコマンドでは、自動的に適切なワークファイルタイプが選択され、このオプションは設定されていても無視されます。 データが内部フォーマットで読み込みおよび書き込みされるときの Natural ワークファイル 1 のワークファイルタイプ。	UNLOAD LOAD SCAN

オプション	説明	利用可能なコマンド
	DEFAULT デフォルトのバイナリワークファイル。 PORTABLE ポータブルワークファイル。 UTF-8 Unicode/UTF-8 でエンコードされたバイナリワークファイル。 UTF-8 は、TRANSFER が指定されている場合にアンロード機能にのみ適用されます。 ［UTF-8］が指定されている場合、［CONVERSION-TABLE］オプション、［SUBSTITUTE］オプションおよび［INCORPORATE-FREE-RULES］オプションは使用できません。 （「ワークファイル」の「ワークファイルフォーマット」も参照してください。） WORKFILETYPE が指定されていない場合、現在のタイプが使用されます。	
ADAFDUWORKFILE	Adabas FDT データがロードされるワークファイル（Natural ワークファイル 5）に割り当てられた完全パス名。	LOAD
PC NEWPC	適用されるのは、Entire Connection がインストールされている場合に限られます。 Entire Connection ワークファイルに対してデータの書き込みまたは読み込みを行います。*file-name* は、Entire Connection ワークファイルに割り当てられている完全パス名を表します。現在のシステム環境でバックスラッシュ（\）セパレータを使用できない場合は、代わりにスラッシュ（/）を使用します。*file-name* を指定しないと、ワークファイル名の入力を求めるプロンプトが Entire Connection から表示されます。 NEWPC が指定されると、既存のワークファイルの内容がアンロードしたデータで上書きされるか、または新しいワークファイルが先頭から埋められます。これに該当しない場合、データが追加されます。 「ワークファイル」の「ワークファイルの割り当て」も参照してください。	UNLOAD LOAD SCAN
PCCOMMANDFILENUMBER または PCCOM または PCCFN	適用されるのは、Entire Connection がインストールされている場合に限られます。 Entire Connection コマンドの処理に使用されるワークファイルの番号を指定します。 デフォルト値はワークファイル 8 の 8 であり、このファイルを Entire Connection ワークファイルとして定義する必要があります。	UNLOAD LOAD SCAN

オプション	説明	利用可能なコマンド
	「ワークファイル」の「*ワークファイルの割り当て*」も参照してください。	

次のセクションでは、*transfer-options* と *internal-format-options* のキーワードおよび変数値（該当する場合）について説明します。

- transfer-options のキーワードの説明
- internal-format-options のキーワードの説明
- USE-FDDM のキーワードの説明

transfer-options のキーワードの説明

TRANSFER キーワードを使用する場合、次のオプションを指定できます。

オプション	説明	利用可能なコマンド
CONVERSION-TABLE	次のいずれかの変換テーブルを使用して、転送フォーマットで処理されたデータを変換します。 SYSTEM-TABLE： 内部的な Natural 変換テーブル。 USER-TABLE： *conversion-program* が指定されている場合のユーザー定義変換テーブル。このプログラムは、ライブラリ SYSOBJH またはその steplib の 1 つに保存されている必要があります。ライブラリ SYSOBJH にあるプログラム例 OTNCONAE および OTNCONEA を参照してください。 *conversion-program* が指定されない場合、NATCONV.INI（[ISO8859_1->EBCDIC] または [EBCDIC->ISO8859_1]）で対応する変換テーブルが使用されます。	UNLOAD LOAD SCAN
SUBSTITUTE	転送フォーマットでのアンロード中に行参照をラベルで置き換えます。 このオプションは、ソースコードの行番号がステートメント参照に使用されている場合にのみ適用されます。その場合、参照された行の行番号および行番号参照はラベルで置き換えられます。ソースはデータベースでは変更されません。	UNLOAD
INCLUDE-LINE-NUMBERS	転送フォーマットでのアンロード中に行番号を転送します。デフォルトでは、Natural オブジェクトの行番号はアンロードされません。	UNLOAD

オプション	説明	利用可能なコマンド
USE-LINE-NUMBER-INCREMENT または USE-LNI	UNLOAD オプション INCLUDE-LINE-NUMBERS が指定されていない場合、Natural ソースオブジェクトの行番号の増分はアンロードされます。デフォルトでは、Natural ソースオブジェクトの行番号の増分はアンロードされません。 LOAD 行番号の増分が転送された場合は、それを使用して Natural ソースオブジェクトの行番号が再構築されます。	UNLOAD LOAD
UPPERCASE-TRANSLATION	転送フォーマットでのロード中にソースコードを大文字に変換します。 デフォルトでは、Natural オブジェクトのソースコードは変換されません。	LOAD
INCORPORATE-FREE-RULES	Predict がインストールされている場合、転送フォーマットでのアンロード中に、マップに連結されている Predict フリールールのソーステキストをマップソースに組み込みます。	UNLOAD
LOAD-CODE-PAGE	Unicode/UTF-8（Universal Transformation Format、8 ビット形式）でエンコードされたオブジェクトソースの変換に使用されるコードページを指定します。 このオプションを使用すると、UTF-8でワークファイルにアンロードされたすべてのオブジェクトソースは、ワークファイルにロードされるときに指定のコードページで変換されます。 *code-page-name* として *CODEPAGE を指定すると、システム変数 *CODEPAGE に割り当てられている値が使用されます（『システム変数』ドキュメントを参照）。 *code-page-name*を指定しないと、ソースオブジェクトは、アンロード時に使用したコードページで変換されます。 ［LOAD-CODE-PAGE］が指定されている場合、［CONVERSION-TABLE］オプションと［UPPERCASE-TRANSLATION］オプションは使用できません。	LOAD LOADALL
DA-FORMAT	データエリアソースのフォーマット変換を指定します。「*設定*」の「*転送オプション*」にある「**Data area format**」を参照してください。	UNLOAD LOAD

internal-format-options のキーワードの説明

internal-format-options を使用する場合、次のオプションを指定できます。

オプション	説明		利用可能なコマンド
XREF	適用されるのは、Predict がインストールされている場合に限られます。 カタログされた Natural オブジェクトの XRef データをロードまたはアンロードします。次の値のいずれかを指定できます。		LOAD UNLOAD
	ON	UNLOAD： カタログ化オブジェクトとそのクロスリファレンスデータ（存在する場合）をアンロードします。 LOAD： クロスリファレンスがワークファイルに存在する場合は、カタログ化オブジェクトとそのクロスリファレンスデータをロードします。	
	OFF	XRef データは処理されません。これがデフォルトです。	
	DOC	LOAD にのみ適用されます。 カタログ化オブジェクトとそのクロスリファレンスデータ（存在する場合）をロードします。ただしそれは、そのオブジェクトの Predict エントリが FDIC システムファイル内に存在する場合に限られます。	
	FORCE	LOAD にのみ適用されます。 カタログ化オブジェクトとそのクロスリファレンスデータをロードします。ただしそれは、クロスリファレンスがワークファイル内に存在し、そのオブジェクトの Predict エントリが FDIC システムファイル内に存在する場合に限られます。	
	SPECIAL	LOAD にのみ適用されます。 カタログ化オブジェクトとそのクロスリファレンスデータ（存在する場合）をロードします。	
DELETEALLOWED	オブジェクトを内部フォーマットでロードするときに、ワークファイルから削除命令を処理します。		LOAD

USE-FDDM のキーワードの説明

UNIX、**OpenVMS**、または **Windows** プラットフォームで **Natural** ライブラリオブジェクトを処理する場合にのみ適用されます。

FDDM システムファイルを処理に使用することを指定します。

FDDM ファイルが NATPARM パラメータファイルでアクティブにされていた場合、デフォルト設定は YES です。

YES または NO の値を指定するときに、以下が適用されます。

値	説明
YES	UNLOAD、FIND および DELETE： パラメータ NATTYPE が V に設定されていると、DDM は、FDDM ファイルにあるライブラリ SYSTEM、またはデータベース ID（DBID）とファイル番号（FNR）で指定されるファイルからのみ処理されます。 パラメータ NATTYPE が * に設定されているか、または NATTYPE がタイプ V 以外の任意の Natural オブジェクトタイプの組み合わせである場合、DDM は処理されません。 LOAD： DDM は、FDDM ファイルにあるライブラリ SYSTEM にロードされます。 *select-clause*の「***Natural** ライブラリオブジェクトと **DDM** 選択*」にある「NATTYPE」の項も参照してください。
NO	UNLOAD、FIND および DELETE： 指定されたライブラリの DDM が処理されます。 LOAD： 指定されたライブラリに DDM がロードされます。

38 ダイレクトコマンドの使用例

- 同一プラットフォームのオブジェクトのアンロード 240
- 異なるプラットフォームのオブジェクトのアンロード 241
- 内部フォーマットでのオブジェクトのロード 242
- 転送フォーマットでのオブジェクトのロード 242

このセクションでは、オブジェクトハンドラのダイレクトコマンドの使用例を示します。

ヒント: 追加例については、オブジェクトハンドラ機能用に生成されたコマンドを表示できます。このコマンドは、ウィザードを使用するときに自動的に表示されます。上級ユーザーモードでは、オブジェクトハンドラコマンド SET ADVANCEDCMD ON を入力するか、オブジェクトハンドラプロファイルでパラメータ Display-Cmd-in-Advanced-Mode を Y（Yes）に設定する（「*プロファイルの設定*」も参照）ことによって、コマンド表示を有効にすることができます。

同一プラットフォームのオブジェクトのアンロード

このセクションでは、ローカルメインフレーム、UNIX、OpenVMS、または Windows のいずれかの環境で、オブジェクトを同一プラットフォームでロードできるようにするために、オブジェクトを内部フォーマットでワークファイルにアンロードする例を示します。

- ライブラリ ABC からすべての Natural プログラミングオブジェクト（ソースオブジェクトのみ）をアンロードします。

```
UNLOAD * LIB ABC OBJTYPE N SCKIND S
```

- ライブラリ ABC からすべての Natural プログラミングオブジェクト（カタログ化オブジェクトのみ）をアンロードします。

```
UNLOAD * LIB ABC OBJTYPE N SCKIND C
```

- ライブラリ ABC からすべての Natural プログラミングオブジェクト（カタログ化オブジェクトとソースオブジェクト）をアンロードします。

```
UNLOAD * LIB ABC OBJTYPE N SCKIND A
```

- ライブラリ ABC からすべての Natural プログラミングオブジェクト（ソースオブジェクトのみ）をアンロードし、ライブラリ ABCNEW にロードします。

```
UNLOAD * LIB ABC OBJTYPE N SCKIND S WITH NEWLIBRARY ABCNEW
```

- メインフレームの場合：名前が EMP で始まりデータベース 88 をポイントしているすべての DDM をアンロードします。

```
UNLOAD EMP* LIB * OBJTYPE D DDMDBID 88
```

- UNIX、OpenVMS、Windows の場合：名前が EMP で始まりデータベース 88 をポイントしているすべての DDM をアンロードします。

```
UNLOAD EMP* LIB * OBJTYPE N NATTYPE V DDMDBID 88
```

- UNIX、OpenVMS、Windowsの場合：名前がEMPで始まるすべてのDDMをライブラリVLIBからアンロードし、ライブラリVLIBNEWにロードします。

```
UNLOAD EMP* LIB VLIB OBJTYPE N NATTYPE V WITH NEWLIBRARY VLIBNEW
```

- すべてのユーザー定義エラーメッセージをライブラリERRLIBからアンロードし、ライブラリNEWERRにロードします。

```
UNLOAD * LIB ERRLIB OBJTYPE E SLKIND A WITH NEWLIBRARY NEWERR
```

- Windowsの場合：すべてのNaturalプログラミングオブジェクト（カタログ化オブジェクトとソースオブジェクト）をライブラリABCからPC上のポータブルワークファイルにアンロードします。

```
UNLOAD * LIB ABC OBJTYPE N WHERE WORKFILE C:\WF1.SAG WORKFILETYPE PORTABLE
```

または

```
UNLOAD * LIB ABC OBJTYPE N WHERE WORK C:\WF1.SAG WFT P
```

異なるプラットフォームのオブジェクトのアンロード

このセクションでは、メインフレームでアンロードしてUNIX、OpenVMS、またはWindows環境でロードするなど、オブジェクトを異なるプラットフォームでロードできるようにするために、オブジェクトを転送フォーマットでワークファイルにアンロードするコマンド例を示します。

- ライブラリABCからすべてのNaturalプログラミングオブジェクト（ソースオブジェクトのみ）をアンロードします。

```
UNLOAD * LIB ABC OBJTYPE N WHERE TRANSFER
```

- すべてのNaturalプログラミングオブジェクト（ソースオブジェクトのみ）とユーザー定義エラーメッセージをライブラリABCからアンロードします。

```
UNLOAD * LIB ABC WHERE TRANSFER
```

- すべてのNaturalプログラミングオブジェクト（ソースオブジェクトのみ）を固定レコード長のライブラリABCからアンロードします。

```
UNLOAD * LIB ABC OBJTYPE N WHERE TRANSFER FIXEDLENGTH
```

内部フォーマットでのオブジェクトのロード

このセクションでは、オブジェクトを内部フォーマットでワークファイルからロードするコマンド例を示します。

- すべてのオブジェクトをライブラリ LIBNEW にロードして、すでに存在しているオブジェクトと置き換えます。

```
LOADALL WITH NEWL LIBNEW WHERE REPLACE ALL
```

- ターゲットライブラリ TGTLIB のすべてのオブジェクトを新しいターゲットライブラリ NEWTGT にロードします。

```
LOAD * LIB TGTLIB WITH NEWLIBRARY NEWTGT
```

- ユーザー定義エラーメッセージ 1000～1500 のみをライブラリ ERRLIB からロードします。

```
LOAD * LIB ERRLIB OBJTYPE E FMNUM 1000 TONUM 1500
```

転送フォーマットでのオブジェクトのロード

このセクションでは、オブジェクトを転送フォーマットでワークファイルからロードするコマンド例を示します。

- すべてのオブジェクトをライブラリ LIBNEW にロードして、すでに存在しているオブジェクトと置き換えます。

```
LOADALL WITH NEWL LIBNEW WHERE TRANSFER REPLACE ALL
```

- ターゲットライブラリ TGTLIB のすべてのオブジェクトを新しいターゲットライブラリ NEWTGT にロードします。

```
LOAD * LIB TGTLIB WITH NEWLIBRARY NEWTGT WHERE TRANSFER
```

39 ナビゲーションと特別な機能のためのコマンド

CUI（キャラクタユーザーインターフェイス）環境におけるオブジェクトハンドラコマンドは、ナビゲーション、トレースファイルの指定などの特別な機能設定を行うことを主な目的としています。

オブジェクトハンドラコマンドは、任意のオブジェクトハンドラ画面のコマンド行に入力されます。オブジェクトハンドラ画面からNaturalシステムコマンドを実行するには、コマンドの前に2つのスラッシュ（//）を入力します。Natural システムコマンドによってオブジェクトハンドラが終了することに注意してください。

手順 39.1. オブジェクトハンドラの［Commands］メニューを表示するには

■ PF10 キー（Cmds）を押します。

または:

オブジェクトハンドラの任意の画面で、コマンド行に次のコマンドを入力します。

```
CMDS
```

オブジェクトハンドラコマンドを以下に示します。キーワードの下線部分は、入力可能な省略形を表します。Sub はサブコマンドを示します。

Command	サブコマンド 1	サブコマンド 2	説明
CANCEL			現在の機能をキャンセルし、オブジェクトハンドラの **Main Menu** を表示します。
CHANGE	WORKPLAN	LIBRARY	管理機能を呼び出し、ワークプランライブラリを変更できる画面を表示します。
CLEAR			マップ内の入力フィールドの現在の内容をデフォルト値にリセットします。
CMDS			［**Commands**］画面が表示されます。

Command	サブコマンド 1	サブコマンド 2	説明
または COMMANDS			
BYE			オブジェクトハンドラを終了します。
EXIT			
QUIT			
.			
FIN			オブジェクトハンドラを終了し、Natural セッションを終了します。
GO	HOME		オブジェクトハンドラ **Main Menu** を表示します。
GO	UNLOAD		アンロード機能を呼び出します。
GO	UNLOAD	END	現在のアンロード機能を終了します。
		ERROR	Natural システムエラーメッセージのアンロード機能を呼び出します。
		EXTERNAL	外部オブジェクトのアンロード機能を呼び出します。
		FDT	FDT のアンロード機能を呼び出します。
		LIBRARY	Natural ライブラリオブジェクトのアンロード機能を呼び出します。
		NCP	Natural コマンドプロセッサソースのアンロード機能を呼び出します。
		RELATED	Natural 関連オブジェクトのアンロード機能を呼び出します。
		SELECTION または LIST	アンロード機能で使用するために SELECTION または LIST ワークプランを入力または選択できる画面を表示します。
GO	LOAD		ロード機能を呼び出します。
GO	LOAD	ALL	ワークファイルに含まれているすべてのオブジェクトのロード機能を呼び出します。
		END	現在のロード機能を終了します。
		ERROR	Natural システムエラーメッセージのロード機能を呼び出します。
		EXTERNAL	外部オブジェクトのロード機能を呼び出します。
		FDT	FDT のロード機能を呼び出します。
		LIBRARY	Natural ライブラリオブジェクトのロード機能を呼び出します。
		NCP	Natural コマンドプロセッサソースのロード機能を呼び出します。
		RELATED	Natural 関連オブジェクトのロード機能を呼び出します。

Command	サブコマンド 1	サブコマンド 2	説明
		SELECTION または LIST	ロード機能で使用するために SELECTION または LIST ワークプランを入力または選択できる画面を表示します。
GO	RESTART		ロードの再スタート機能で使用するためにファイルを指定できる画面を表示します。
GO	SCAN		スキャン機能を呼び出します。
GO	SCAN	ALL	ワークファイルに含まれているすべてのオブジェクトのスキャン機能を呼び出します。
		END	現在のスキャン機能を終了します。
		ERROR	Natural システムエラーメッセージのスキャン機能を呼び出します。
		EXTERNAL	外部オブジェクトのスキャン機能を呼び出します。
		FDT	FDT のスキャン機能を呼び出します。
		LIBRARY	Natural ライブラリオブジェクトのスキャン機能を呼び出します。
		NCP	Natural コマンドプロセッサソースのスキャン機能を呼び出します。
		RELATED	Natural 関連オブジェクトのスキャン機能を呼び出します。
		SELECTION または LIST	スキャン機能で使用するために SELECTION または LIST ワークプランを入力または選択できる画面を表示します。
GO	ADMIN		管理機能を呼び出します。
GO	ADMIN	CHANGE	ワークプランライブラリを変更できる画面を表示します。
		CREATE	ワークプランを作成できるメニューを開きます。
		LIST	ワークプランライブラリで使用可能なワークプランのリストを生成します。
		GO	VIEW 表示機能を呼び出します。
GO	VIEW	ERROR	Natural システムエラーメッセージの表示機能を呼び出します。
		FDT	FDT の表示機能を呼び出します。

Command	サブコマンド 1	サブコマンド 2	説明
		LIBRARY	Natural ライブラリオブジェクトの表示機能を呼び出します。
		NCP	Natural コマンドプロセッサソースの表示機能を呼び出します。
GO	FIND		検索機能を呼び出します。
GO	FIND	ERROR	Natural システムエラーメッセージの検索機能を呼び出します。
		FDT	FDT の検索機能を呼び出します。
		LIBRARY	Natural ライブラリオブジェクトの検索機能を呼び出します。
		NCP	Natural コマンドプロセッサソースの検索機能を呼び出します。
		SELECTION または LIST	検索機能で使用するために SELECTION または LIST ワークプランを入力または選択できる画面を表示します。
HELP			オブジェクトハンドラのヘルプ機能を呼び出します。
INIT			オブジェクトハンドラユーティリティを再初期化します。
READ	PROFILE		テキストオブジェクト PROFILE に定義されているように、オブジェクトハンドラの設定を更新します（「プロファイルの設定」も参照）。
SET	ADVANCEDCMD	ON	オブジェクトハンドラで生成されたコマンドの上級ユーザーモードおよびコンパクト入力モードでの表示を有効にします。
		OFF	オブジェクトハンドラで生成されたコマンドの上級ユーザーモードおよびコンパクト入力モードでの表示を無効にします。
	EXECUTIONMSG	ON	処理ステータスが表示されるウィンドウを有効にします。
		OFF	処理ステータスが表示されるウィンドウを無効にします。
	FREE	ON	フリーフォーマット編集を有効にします。
		OFF	フリーフォーマット編集を無効にします。
	Input-Mode, IM	W	［ウィザードモード］を選択します。
		A	［上級ユーザーモード］を選択します。
		C	［コンパクトモード］を選択します。
	TRACE	ON	トレースモードを有効にします。オブジェクトハンドラの各操作が画面に表示されます。
		OFF	トレースモードを無効にします。

Command	サブコマンド 1	サブコマンド 2	説明
		WORKFILE	トレースモードを有効にします。オブジェクトハンドラの各操作がワークファイル 10 に書き込まれます。
	TRACEFILE		トレースファイル（ワークファイル10）の名前を指定できる画面を表示します。
SETTINGS			アンロード、ロードまたはスキャン設定を指定できる画面を表示します。
SHOW または DISPLAY	LAST	MESSAGE	オブジェクトハンドラの処理インターフェイスによって発行された最後のインターフェイスのリターンコードとメッセージが表示されます。
		RESULT	オブジェクトハンドラの処理インターフェイスによって発行された最新の結果が表示されます。
	PROFILE		オブジェクトハンドラプロファイルを表示または変更します。
	REPORT		最後に作成されたレポートが表示されます。
	STATISTICS		処理されたオブジェクトの統計情報が表示されます。
	STATUS		オブジェクトハンドラの現在のステータス（グローバル変数の内容）が表示されます。
	TRACE	FILE	トレースファイル（ワークファイル10）の内容が表示されます。

40 バッチのコンディションコードとユーザー出口ルーチン

- バッチで返されるコンディションコード 250
- ユーザー出口ルーチンの適用 250
- 使用可能なユーザー出口ルーチン 251

このセクションでは、バッチモードのオブジェクトハンドラ機能に対して返されるコンディションコードおよび機能処理に使用できるユーザー出口ルーチンについて説明します。

バッチで返されるコンディションコード

バッチモードのオブジェクトハンドラ処理は、次のいずれかのコンディションコードで終了します。

コンディションコード	説明
0	オブジェクトハンドラ処理は正常終了しました。
30	内部的なオブジェクトハンドラエラーが発生しました。
40	オブジェクトハンドラコマンドでエラーが検出されました。
50	オブジェクトハンドラ処理中にエラーが発生しました。
60	オブジェクトハンドラ処理中に Natural Security エラーが発生しました。
99	オブジェクトハンドラ処理中に Natural エラーが発生しました。

ユーザー出口ルーチンの適用

オブジェクトハンドラのユーザー出口ルーチンは、Natural システムライブラリ SYSOBJH にソースオブジェクトとして提供されます。これらのソースオブジェクトには SRC-EX*nn* という名前が付いており、*nn* はユーザー出口ルーチンの番号を表します。

手順 40.1. ユーザー出口ルーチンを起動するには

- Natural システムライブラリ SYSOBJH 内の `OBJHEX`*nn* という名前で、ソースオブジェクト SRC-EX*nn* に対し `CATALOG` または `STOW` を実行します。

  ユーザー出口ルーチンのソースオブジェクト（要件に応じて変更可能）およびカタログ化オブジェクトが更新インストールで上書きされないように、別の名前が使用されます。

  ユーザー出口ルーチンの詳細については、ライブラリ SYSOBJH にある SRC-EX*nn* のソースオブジェクトを参照してください。

使用可能なユーザー出口ルーチン

次のユーザー出口ルーチンを使用することができます。

- 障害処理のための OBJHEX01
- オブジェクト拒否のための OBJHEX02
- デフォルトオプション値のための OBJHEX03
- Natural オブジェクトタイプ統計の OBJHEX04

障害処理のための OBJHEX01

コンディションコードがバッチモードで0（ゼロ）より大きい値に設定されると、オブジェクトハンドラが処理を停止する前に、ユーザー出口ルーチン OBJHEX01（使用可能な場合）が呼び出されます。このユーザー出口ルーチンで、オブジェクトハンドラ処理の続行または終了を指定できます。終了する場合には、コンディションコードを変更できます。詳細については、Natural システムライブラリ SYSOBJH のユーザー出口ルーチン SRC-EX01 のソースを参照してください。

オブジェクト拒否のための OBJHEX02

オブジェクトハンドラのロード機能がバッチモード（コンディションコード0）またはオンラインコマンドモードで正常に実行されても、1つまたは複数のオブジェクトがロード中に拒否された（例：置換されなかった）場合、オブジェクトハンドラが処理を停止する前に、ユーザー出口ルーチン OBJHEX02（使用可能な場合）が呼び出されます。OBJHEX02 で、オブジェクトハンドラ処理の続行または終了を指定できます。終了する場合には、コンディションコードを変更できます。詳細については、Natural システムライブラリ SYSOBJH のユーザー出口ルーチン SRC-EX02 のソースを参照してください。

デフォルトオプション値のための OBJHEX03

ユーザー出口ルーチン OBJHEX03 を適用して、オブジェクトハンドラコマンドの処理に対してデフォルトオプションを設定できます。このユーザー出口ルーチンは、オブジェクトハンドラコマンドが処理される前に呼び出されます。詳細については、Natural システムライブラリ SYSOBJH のユーザー出口ルーチン SRC-EX03 のソースを参照してください。

Natural オブジェクトタイプ統計の OBJHEX04

オブジェクトハンドラのアンロード、ロード、またはスキャン機能がコマンドモードで正常に実行された場合、ユーザー出口ルーチン OBJHEX04（使用可能な場合）が呼び出されます。処理済み、置換済み、未置換、または拒否済みのカタログ化オブジェクトおよびソースの各Natural オブジェクトタイプの統計を提供します。OBJHEX04 で、オブジェクトハンドラ処理の続行または終了をバッチモードで指定できます。終了する場合には、コンディションコードを変更できます。詳細については、Natural システムライブラリ SYSOBJH のユーザー出口ルーチン SRC-EX04 のソースを参照してください。

41 Tools

- ステータス 254
- 最新の結果 254
- 「トレース」 254
- レポート 255

オブジェクトハンドラには、ステータス情報とレポートを表示し、トレース設定のチェックと修正を行うための特殊機能が用意されています。

ステータス

現在使用されているオブジェクトハンドラ機能、ユーザー環境、ワークプランライブラリ、および以下で説明しているトレースオプションの設定を表示します。

手順 41.1. ステータスを表示するには

■ 任意のオブジェクトハンドラ画面で、コマンド行に次のコマンドを入力します。

```
SHOW STATUS
```

「*ダイレクトコマンド*」セクションの「*ナビゲーションと特別な機能のためのコマンド*」にある SHOW コマンドの説明も参照してください。

最新の結果

オブジェクトハンドラの処理インターフェイスによって発行された最後の内部コマンドおよび有効なリターンコードとメッセージを表示します。

手順 41.2. 最新の結果を表示するには

■ 任意のオブジェクトハンドラ画面で、コマンド行に次のコマンドを入力します。

```
SHOW LAST RESULT
```

「*ダイレクトコマンド*」セクションの「*ナビゲーションと特別な機能のためのコマンド*」にある SHOW コマンドの説明も参照してください。

「トレース」

トレース機能を有効または無効にします。トレース機能により内部のオブジェクトハンドラのプログラムフローが記録され、エラー診断用の制御情報が提供されます。トレースオプションはデフォルトでオフに設定されています。

手順 41.3. 設定を変更するには

- 「*ダイレクトコマンド*」セクションの「*ナビゲーションと特別な機能のためのコマンド*」の説明に従って、コマンド `SET TRACE` を使用します。

レポート

ロード、アンロード、またはスキャンされたオブジェクトをリスト表示し、処理を中断する可能性のあるエラーを記録します。「*設定*」セクションの「*ワークファイルオプション*」も参照してください。レポートオプションはデフォルトで設定されており、アンロード、ロード、またはスキャン機能の実行後に表示されます。

手順 41.4. 最新レポートファイルの内容を表示するには

- 任意のオブジェクトハンドラ画面で、コマンド行に次のコマンドを入力します。

  ```
  SHOW REPORT
  ```

  「*ダイレクトコマンド*」セクションの「*ナビゲーションと特別な機能のためのコマンド*」にある `SHOW` コマンドの説明も参照してください。

42 プロファイル設定

- PF キー 259
- 行コマンド 259
- プロファイルパラメータ 260

Naturalには、現在のオブジェクトハンドラユーティリティ環境のデフォルト設定をカスタマイズするオプションが用意されています。このために、Natural システムライブラリ SYSOBJH にはテキストオブジェクト OBJHPROF が提供されています。OBJHPROF は、フラグやオプションの環境固有のデフォルト値を指定するために使用します。これらは、対応するオブジェクトハンドラ画面を入力するときに表示されます。

手順 42.1. 個々のプロファイル設定を有効にするには

- オンラインモードで、オブジェクトハンドラ内部コマンド PROFILE（または SHOW PROFILE）を入力します。

  このコマンドは、プロファイルメンテナンスツールを呼び出して、次のことを実行します。

  - 一般またはユーザー固有のプロファイルパラメータと現在の値をマップとして表示します。
  - オブジェクトハンドラプロファイルが存在しない場合、内部的に使用されるデフォルト値を持つ新しいオブジェクトハンドラプロファイルを作成します。
  - （Natural Security によって制御される）オブジェクトハンドラプロファイルのプロファイルパラメータに対する一般的な設定を変更できるようにします。
  - （Natural Security によって制御される）オブジェクトハンドラプロファイルのプロファイルパラメータに対するユーザー固有の設定を変更できるようにします。
  - 各プロファイルパラメータの説明とヘルプ情報を提供します。

PF4キーを入力してユーザープロファイル設定を切り替えます。*行コマンド*を使用して、ユーザー固有のプロファイルからパラメータを追加、修正、または削除できます。

注意:

1. 新しいパラメータが追加されると、プロファイルメンテナンスツールの内部コマンド UPDATE によって変更がオブジェクトハンドラプロファイルに取り込まれます。詳細については、**Help** を呼び出してください。
2. オブジェクトハンドラプロファイル自体はOBJHPROF と名付けられています。これは、ライブラリ SYSOBJH にあります。オブジェクトハンドラプロファイルの更新に使用されるデフォルトプロファイルは、OBJHDEFP という名前です。これは、ライブラリ SYSOBJH にもあります。

手順 42.2. 個々のプロファイル設定を無効にするには

- テキストオブジェクト OBJHPROF をライブラリ SYSOBJH から削除します。

PF キー

次の PF キーが有効です。

PF キー	説明
PF1 (**Help**)	一般的なヘルプまたはコンテキスト依存ヘルプを呼び出します。
PF2 (**Print**)	現在のプロファイル設定を出力します。
PF3 (**Exit**)	プロファイルデータを保存し、プロファイルメンテナンスツールを終了します。
PF4 (**Gener/User**)	一般的な（**Gener**）プロファイル設定とユーザー固有の（**User**）プロファイル設定を切り替えます。
PF5 (**Modif**)	別のメニューで（カーソルで選択してマークされた）パラメータを変更します。
PF6 (**--**)	パラメータリストの先頭までスクロールします。
PF7 (**-**)	1 ページ分戻る方向にスクロールします。
PF8 (**+**)	1 ページ分進む方向にスクロールします。
PF9 (**++**)	パラメータリストの末尾までスクロールします。
PF10 (**ALLDU/ALLAU**)	すべてのパラメータに対して、行コマンド DU または AU を挿入します。
PF12 (**Canc**)	プロファイルメンテナンスツールを終了します。

行コマンド

次の行コマンドを使用できます。

行コマンド	説明
MO	拡張モードでパラメータ値を変更します。
AU	ユーザープロファイルにエントリを追加します
DU	ユーザープロファイルからエントリを削除します
DI	パラメータの説明を表示します
HE	パラメータに関するヘルプ情報を表示します

プロファイルパラメータ

下の表に、オブジェクトハンドラプロファイル OBJHPROF に含まれるパラメータ、入力可能な値、およびパラメータが適用されるオブジェクトハンドラ機能を示します。また、パラメータの簡単な説明または『オブジェクトハンドラ』ドキュメントの該当する参照セクションも示します。デフォルトのパラメータ値は、下線付きで示しています。

パラメータ	指定可能な値	関数	説明／ ドキュメントのセクション
Input-Mode	W	アンロード ロード スキャン	*ウィザード*
	A		*上級ユーザー*
	C		*コンパクトモード*
Display-Cmd-in-Advanced-Mode	N または Y	アンロード ロード スキャン	上級ユーザーモードで実行される機能用に生成されたオブジェクトハンドラコマンドを表示します。
Display-ExecutionMsg	N または Y	アンロード ロード スキャン	処理ステータスが表示されるウィンドウを有効にします。
Display-Statistics	N または Y	アンロード ロード スキャン	機能の実行後に処理されたオブジェクトに関する統計を表示します。（ダイレクトコマンド SHOW STATISTICS と同等）。
Workplan-Library	WORKPLAN または 他の任意のワークプランライブラリ	アンロード ロード スキャン 管理	「*ワークプラン*」および「*ワークプランライブラリの変更*」（「*管理*」）
Workplan-Library-DBID	0（現在の FNAT/FUSER） または 他の任意の Adabas データベース ID（DBID）	アンロード ロード スキャン 管理	*ワークプランライブラリの変更*
Workplan-Library-FNR	0（現在の FNAT/FUSER） または 他の任意の Adabas ファイル番号（FNR）	アンロード ロード スキャン 管理	*ワークプランライブラリの変更*

パラメータ	指定可能な値	関数	説明／ ドキュメントのセクション
TRACE	N または Y	アンロード ロード スキャン	「ツール」の「トレース」
TRACE-TARGET	S（Screen） または W（Work file）	アンロード ロード スキャン	「トレース」
Option-Replace	N または Y または O（Obsolete） E（Except）	ロード	「設定」の「置換オプション」
Option-TRANSFER-FORMAT	N または Y	アンロード ロード スキャン	「ワークファイル」の「ワークファイルフォーマット」
Option-Use-PC-Work-File	N または Y	アンロード ロード スキャン	以下を参照してください。 「その他のオプションの設定」の「*Use PC File*」
Option-TR-INCLUDE-LINE-NUMBERS	N または Y	アンロード	の「*Include line numbers*」 「転送オプション」（『設定』） または 「転送オプション」（ダイレクトコマンド）
Option-TR-LINE-NUMBER-INCREMENT	N または Y	アンロード ロード	「転送オプション」の 「**USE-LINE-NUMBER-INCREMENT**」（ダイレクトコマンド）
Option-TR-SUBSTITUTE	N または Y	アンロード	**Substitute line references** 「転送オプション」（『設定』） または **SUBSTITUTE** 「転送オプション」（『ダイレクトコマンド』）
Option-TR-TRANSLATE-TO-UPPER	N または Y	ロード	**Translate to upper case** 「転送オプション」（『設定』） または 「転送オプション」の 「**UPPERCASE-TRANSLATION**」（『ダイレクトコマンド』）
Option-TR-USE-CONVERSION-TABLE	N または	アンロード	**Use conversion table** 「転送オプション」（『設定』）

パラメータ	指定可能な値	関数	説明／ ドキュメントのセクション
	S（System table） または U（User table）	ロード	または 「*転送オプション*」の 「**CONVERSION-TABLE**」（『*ダイレクトコマンド*』）
Option-TR-CONV-TABLE-NAME-LOAD	OTNCONEA または ユーザー記述サブプログラム	ロード	**Use conversion table** 「*転送オプション*」（『*設定*』） または 「*転送オプション*」の 「**CONVERSION-TABLE**」（『*ダイレクトコマンド*』）
Option-TR-CONV-TABLE-NAME-UNLD	OTNCONAE または ユーザー記述サブプログラム	アンロード	**Use conversion table** 「*転送オプション*」（『*設定*』） または 「*転送オプション*」の 「**CONVERSION-TABLE**」（『*ダイレクトコマンド*』）
Option-TR-DA-FORMAT	N または *	アンロード ロード	**Data area format** 「*転送オプション*」 または 「*転送オプション*」の「**DA-FORMAT**」（『*ダイレクトコマンド*』）
Option-TR-UNICODE-WORK-FILE	N または Y	UNLOAD	**Unicode work file** 「*設定画面のフィールド*」
Option-TR-LOAD-CODE-PAGE	Code page name または *CODEPAGEY	LOAD	**Use load code page** 「*転送オプション*」（『*設定*』）
Option-Write-Report	N または Y	アンロード ロード スキャン	「*ツール*」の「*レポート*」 **Write report** 「*ワークファイルオプションとレポートオプション*」
Default-Report-Direct-Command	B または N または Y	アンロード ロード スキャン	ダイレクトコマンドが実行され、オブジェクトハンドラコマンドにレポートオプションが指定されていない場合に書き込まれるレポートの種類を定義します。 Bバッチレポートを作成します。レポートデータは出力デバイスに直接書き込まれます。オプション BATCHREPORT に対応します。 Yワークファイル4を使用してレポートを作成するか、メインフレームで作業している場合はテキストオブジェクト

パラメータ	指定可能な値	関数	説明／ ドキュメントのセクション
			を使用してレポートを作成します。オプション REPORT に対応します。 Nレポートを作成しないでください。オプション NOREPORT に対応します。
Default-Report-Option-1	A または E または S	アンロード ロード スキャン 削除	ダイレクトコマンドが実行されたときに、オブジェクトハンドラコマンドでオプション REPORT-OPTION-1 が指定されていない場合に、レポートが書き込まれるオプションを定義します。有効な値については、「REPORT-OPTION-1」を参照してください。
Default-Report-Format	S または T	アンロード	ダイレクトコマンドが実行されたときに、オブジェクトハンドラコマンドでオプション REPORT-FORMAT が指定されていない場合に、書き込まれるレポートのフォーマットを定義します。有効な値については、「REPORT-FORMAT」を参照してください。
Default-Report-Mode	S または L	アンロード ロード スキャン	ダイレクトコマンドが実行され、オブジェクトハンドラコマンドでオプション REPORT-MODE が指定されていない場合に使用されるレポートモードを定義します。有効な値については、「REPORT-MODE」を参照してください。
Option-Write-Restart-Info	Y または N	ロード	**Write restart information**「ワークファイルオプションとレポートオプション」
USE-OPTION-WORKPLAN	N または Y	アンロード ロード スキャン	「ワークプラン」
OPTION-WORKPLAN-Name	OPTIONWP または タイプ OPTION の他の任意のワークプラン	アンロード ロード スキャン	「ワークプラン」
USE-PARAMETER-WORKPLAN	N または Y	アンロード ロード	「ワークプラン」
PARAMETER-WORKPLAN-Name	PARAWPLN または	アンロード	「ワークプラン」

パラメータ	指定可能な値	関数	説明／ ドキュメントのセクション
	タイプ PARAMETER の他の任意のワークプラン	ロード	
WORK-FILE-1-Name	ワークファイル 1 に割り当てられた完全パス名。	アンロード ロード スキャン	*ワークファイル*
Report-File-Name	ワークファイル 4 に割り当てられた完全パス名。	アンロード ロード スキャン	**Write report**「*ワークファイルオプションとレポートオプション*」
Restart-File-Name	ワークファイル 6 に割り当てられた完全パス名。	ロード	**Write restart information**「*ワークファイルオプションとレポートオプション*」
Trace-File-Name	ワークファイル 10 に割り当てられた完全パス名。	すべての機能	「*ツール*」の「*トレース*」
DELETE-TEMPORARY-REPORT-TEXT	N または Y	アンロード、ロード、スキャン	メインフレームのみで有効：Y＝レポートの一時的なテキストオブジェクトは、オブジェクトハンドラが終了したときに削除されます（デフォルト値）N＝レポートの一時的なテキストオブジェクトは、オブジェクトハンドラが終了しても削除されません

43 NATUNLD/NATLOAD および SYSTRANS からオブジェクトハンドラへの移行

- 個々のコマンドの変換 266
- OBJHAPI を使用した SYSTRANS コマンドの処理 268
- サポートされていない SYSTRANS オプション 268

古いユーティリティ NATUNLD/NATLOAD および SYSTRANS からオブジェクトハンドラに移行するには、このセクションで説明する 2 つの方法を使用できます。

個々のコマンドの変換

移行用に提供されているオブジェクトハンドラコマンドを使用することにより、NATUNLD/NATLOAD または SYSTRANS ダイレクトコマンドを対応するオブジェクトハンドラコマンドに変換できます。これらの移行コマンドにより、古いユーティリティにより使用されるコマンド構文が、オブジェクトハンドラにより使用されるコマンド構文に自動的に変換されます。

手順 43.1. 単一のコマンドを変換するには

1 次のいずれかのオブジェクトハンドラダイレクトコマンドを使用します。

```
NATUNLD
```

続いて NATUNLD ダイレクトコマンドを入力します。

または:

```
NATLOAD
```

続いて NATLOAD ダイレクトコマンドを入力します。

または:

```
SYSTRANS
```

続いて SYSTRANS ダイレクトコマンドを入力します。

指定したユーティリティコマンドが、対応するオブジェクトハンドラコマンドに変換されます。

2 オブジェクトハンドラの後続のコマンドを、それぞれユーティリティ NATUNLD、NATLOAD、または SYSTRANS に適用される構文で指定します。

このユーティリティの構文は、オブジェクトハンドラセッションの期間中は有効です。

NATUNLD コマンドの例

以下に、2 つの連続する NATUNLD ユーティリティコマンドと、それに対応するオブジェクトハンドラコマンドの例を示します。

古い NATUNLD コマンド：	NATUNLD ALL * FM LIB1 TO LIB2
	ALL PG* FM LIB2
新しいオブジェクトハンドラコマンド：	SYSOBJH NATUNLD ALL * FM LIB1 TO LIB2
NATUNLD 構文での後続のオブジェクトハンドラコマンド：	ALL PG* FM LIB2

SYSTRANS コマンドの例：

以下に、2 つの連続する SYSTRANS ユーティリティコマンドと、それに対応するオブジェクトハンドラコマンドの例を示します。

古い SYSTRANS コマンド：	TRANSCMD EXECUTE UNLOAD N FROM LIB1 NAME ETID
	END
新しいオブジェクトハンドラコマンド：	SYSOBJH SYSTRANS EXECUTE UNLOAD N FROM LIB1 NAME ETID END
SYSTRANS 構文での後続のオブジェクトハンドラコマンド：	END

バッチ処理での **SYSTRANS** の例：

以下に、マップ入力データおよび対応するオブジェクトハンドラコマンドと入力データを使用してバッチで SYSTRANS ユーティリティコマンドを処理する例を示します。

古い SYSTRANS バッチシーケンス：

```
SYSTRANS
U
N,N,N,Y,N,N,N,N
N
SRCLIB1,PGM1,*,TGTLIB1
```

新しいオブジェクトハンドラバッチシーケンス：

```
SYSOBJH SYSTRANS
U
N,N,N,Y,N,N,N,N
N
SRCLIB1,PGM1,*,TGTLIB1
```

OBJHAPI を使用した SYSTRANS コマンドの処理

（Natural システムライブラリ SYSOBJH 内に提供されている）OBJHAPI アプリケーションプログラミングインターフェイスを使用すると、SYSTRANS ユーティリティの構文でオブジェクトハンドラコマンドを実行できます。

この目的で OBJHAPI を使用する場合、OBJHAPI を呼び出すプログラム内でパラメータ P-EXTENSIONS-EXEC-SYSTRANS-CMD を指定する必要があります。詳細については、ライブラリ SYSOBJH 内に提供されているプログラム例 DOC-API を参照してください。

サポートされていない SYSTRANS オプション

オブジェクトハンドラは、次の SYSTRANS ダイレクトコマンドオプションをサポートしていません。WORK-FILE-INPUT、SPECIAL-CONVERSION、RULE-LOAD、および UNLOAD-RULES

VI SYSERR ユーティリティ

Natural アプリケーションを開発する場合、エラーメッセージや情報メッセージと Natural コードを区別して、別々に管理できます。これにより、メッセージの標準化、さまざまなメッセージのメッセージ範囲の事前定義、メッセージの他の言語への翻訳、メッセージの詳細を説明する長文テキストのメッセージへの付加などを簡単に行うことができます。

SYSERR ユーティリティには、アプリケーション固有のメッセージを書き込むオプションがあります。また、SYSERR ユーティリティを使用して、既存の Natural システムメッセージをカスタマイズすることもできます。

メッセージの全般的な情報

SYSERR 呼び出し

機能

パラメータ

ダイレクトコマンド

大文字変換 **- ERRUPPER**

文字置換 **- ERRCHAR**

メッセージファイルとテキストファイルの生成

さまざまなライブラリのメッセージの管理

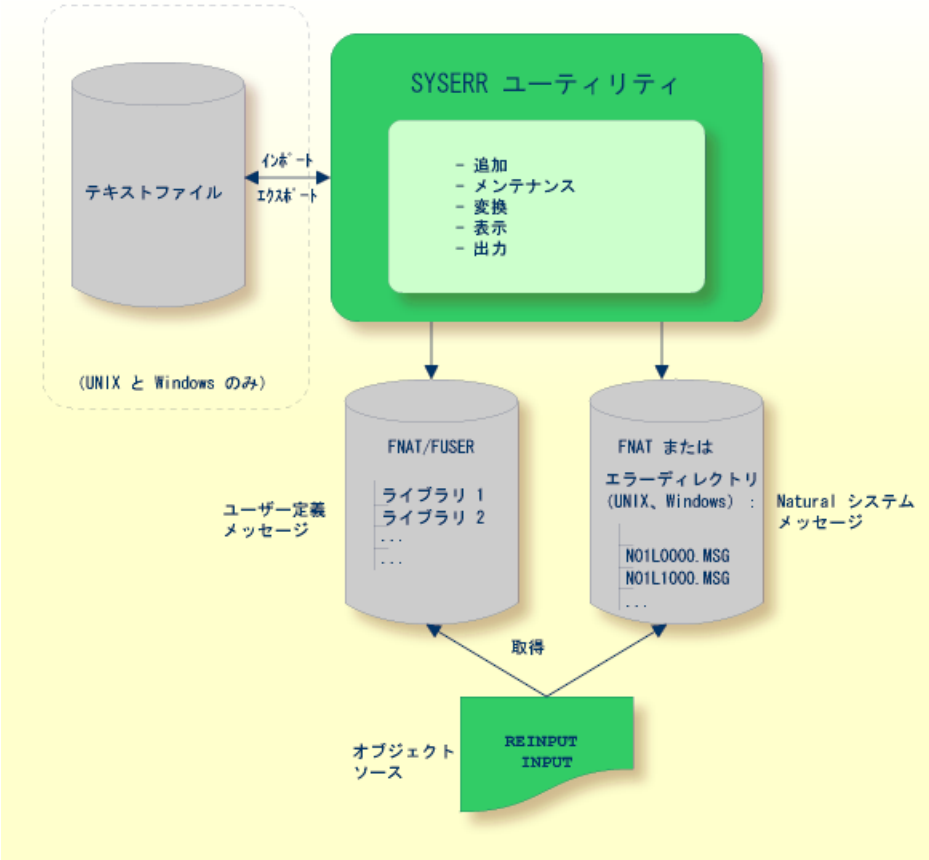
アプリケーションプログラミングインターフェイス **USR0020P**

44 メッセージの全般的な情報

- メッセージタイプ 272
- メッセージの言語 273
- メッセージの発行 274
- Natural システムショートメッセージの取得 275
- ユーザー定義ショートメッセージの取得 275
- メッセージ情報の取得 276

このセクションでは、SYSERRユーティリティを使用して管理できるメッセージとメッセージ言語のタイプ、およびNaturalシステム環境でのメッセージの発行および取得方法について説明します。

次の図に、SYSERR ユーティリティの機能と Natural でのメッセージの処理を示します。



メッセージタイプ

メッセージには、次の 2 つのタイプがあります。Natural（システム）メッセージとユーザー定義メッセージ：

Natural システムメッセージは、Natural ニュークリアスおよび Natural ユーティリティによって発行されます。Natural システムメッセージは Software AG によって提供され、Natural Err ディレクトリにメッセージファイルとして保存されます。Natural システムメッセージは、NAT0230 のように、NAT で始まり 4 桁の数字が続きます。

ユーザー定義メッセージは、ユーザーが作成したアプリケーションによって発行されます。ユーザー定義メッセージは、システムファイル FUSER または FNAT のライブラリ（SYS- ライブラリを含む）にメッセージファイルとして保存されます。

メッセージは、他の言語に翻訳できます。各言語は、個別のメッセージファイルに保存されます。各ライブラリおよびメッセージファイルには、最大 9999 のメッセージを保存できます。

メッセージテキストには、次の 4 つのタイプがあります。

- Natural システムショートメッセージ
- Natural システムロングメッセージ
- ユーザー定義ショートメッセージ
- ユーザー定義ロングメッセージ

ショートメッセージは1行のメッセージで、対応するエラー状況が発生すると、メッセージ行に表示されます。

ロングメッセージは対応するショートメッセージの詳細な説明で、問題の解決方法も含まれています。

注意: Naturalシステムメッセージを変更すると、誤ったメッセージになったり、Naturalの新しいバージョンのリリース時に変更が失われたりする可能性があります。

メッセージの言語

メッセージは、最大 60 の言語で作成できます。『システム変数』ドキュメントのシステム変数「*LANGUAGE」を参照してください。

次のルールおよび制限が適用されます。

- Naturalシステムショートメッセージは、最初に英語で入力する必要があります。その後、他の言語に翻訳できます。
- Naturalシステムロングメッセージは、英語で入力できますが、他の言語に翻訳できません。
- ユーザー定義ショートメッセージは、任意の言語で入力でき、その後、他の言語に翻訳できます。
- ユーザー定義ロングメッセージは、任意の言語で入力できますが、同じ言語の対応するショートメッセージがある必要があります。

メッセージの発行

このセクションでは、Natural プログラムで Natural システムショートメッセージまたはユーザー定義ショートメッセージを発行するために使用する Natural ステートメント INPUT および REINPUT について説明します。

手順 44.1. プログラムで Natural システムショートメッセージを発行するには

■ 次のいずれかの Natural ステートメントを指定します。

```
INPUT WITH TEXT *-nnnn ' '
```

または

```
REINPUT WITH TEXT *-nnnn
```

nnnn は、要求されたメッセージの番号です（先頭のゼロは省略可能）。

手順 44.2. プログラムでユーザー定義ショートメッセージを発行するには

■ 次のいずれかの Natural ステートメントを指定します。

```
INPUT WITH TEXT *nnnn ' '
```

または

```
REINPUT WITH TEXT *nnnn
```

nnnn は、要求されたメッセージの番号です（先頭のゼロは省略可能）。

メッセージテキストのダイナミック置換

メッセージテキストには、表記 :*n*: で識別される変数部分を含めることができます。*n* はオカレンス 1～7 を表します。変数部分は、ランタイムの値に置き換えられます。

詳細については、「*INPUT 構文 1 - ダイナミック画面レイアウトの指定*」セクションの「*operand3*」または『*ステートメント*』ドキュメントの「*REINPUT*」セクションの「*operand3*」を参照してください。

Natural システムショートメッセージの取得

プログラムでNaturalシステムショートメッセージが参照されている場合、Naturalでは、Natural Err ディレクトリ内の要求されたメッセージ番号が次の順序で検索されます。

1. システム変数 `*LANGUAGE` で設定されている現在の言語コード
2. 言語コード 1（英語）

上記のいずれも見つからない場合は、存在しないメッセージがプログラムで参照されています。表示されるのは、NAT0230 など、接頭辞が `NAT` のメッセージ番号のみです。

ユーザー定義ショートメッセージの取得

プログラムでユーザー定義ショートメッセージが参照されている場合、Naturalでは、最初に、システム変数 `*LANGUAGE` で設定されている現在の言語コードで要求されたメッセージ番号 *nnnn* が検索されます（『システム変数』ドキュメントを参照）。メッセージがない場合は、言語コード 1（英語）で要求されたメッセージ番号 *nnnn* が検索されます。ここにもメッセージがない場合は、言語コード 1 で要求されたメッセージ番号 *n*000（*n* は要求されたメッセージ番号の最初の数字）が検索されます。

この3段階の検索は、最初に現在のライブラリで実行されます。ここで何も見つからなかった場合は、対応するメッセージが見つかるまで、他のライブラリが同様に検索されます。

検索されるライブラリの順序は、次のとおりです。

1. システム変数 `*LIBRARY-ID` で設定されている現在のライブラリ。
2. steplib。Natural Security がインストールされている場合は、現在のライブラリの Natural Security プロファイルで steplib が指定される順序。
3. システム変数 `*STEPLIB` で設定されているデフォルトの steplib。
4. システムファイル FUSER（*）のライブラリ SYSTEM。
5. システムファイル FNAT（*）のライブラリ SYSTEM。

（*）現在のライブラリのファイル名が SYS で始まる場合は、SYSTEM FUSER の前に SYSTEM FNAT が検索されます。

メッセージ情報の取得

ショートメッセージが表示された場合、問題状況についての追加情報を検索できます。

- システムコマンド HELP を使用すると、Natural システムロングメッセージまたはユーザー定義ロングメッセージを表示できます。
- システムコマンド LASTMSG を使用すると、最後に表示されたメッセージのショートテストおよびエラー状況に関する追加情報をリストできます。表示される情報には、最後のメッセージの前に表示された可能性がある関連エラーメッセージが含まれます。

これらのコマンドについては、『システムコマンド』ドキュメントを参照してください。

45 SYSERR 呼び出し

手順 45.1. SYSERR ユーティリティを呼び出すには

■ 次のシステムコマンドを入力します。

```
SYSERR
```

SYSERR ユーティリティのメインメニューが表示されます。

```
16:10:42              ***** 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________
Please enter code.
Command ===>
Enter-PF1---PF2---PF3---PF4---PF5---PF6---PF7---PF8---PF9---PF10--PF11--PF12---
      Help        Exit                                                   Canc
```

SYSERR ユーティリティのメインメニューから、メッセージを作成および管理するすべての SYSERR 機能を実行できます。各機能については、「*機能*」を参照してください。機能とともに適用されるパラメータの概要については、「*パラメータ*」を参照してください。パラメータの使用に適用される制限については、「*機能*」の各機能の説明を参照してください。

SYSERR ユーティリティでは、詳細なオンラインヘルプを利用できます。フィールド固有のヘルプ情報を表示するには、関連するフィールドに疑問符を入力して ENTER キーを押すか、カーソルを目的のフィールドに置いて PF1 キーを押します。

46 機能

- メッセージの追加 280
- メッセージの削除 283
- メッセージの表示 284
- メッセージの変更 285
- メッセージの出力 287
- メッセージのスキャン 289
- リストからのメッセージの選択 291
- メッセージの他の言語への翻訳 293

SYSERRユーティリティ機能を呼び出すには、SYSERRメインメニューの入力フィールドに、必要な機能に対応するコードおよび1つ以上のパラメータを入力します。このセクションでは、メニューに表示される機能と、各機能に対して指定できるパラメータについて説明します。パラメータの一般的な使用方法については、「パラメータ」を参照してください。

メッセージの追加

手順 46.1. 新しいメッセージを追加するには

1 SYSERR ユーティリティでは、デフォルトで大文字と小文字が区別されます。作成されるメッセージで大文字と小文字の変換を行う場合は、次の端末コマンドを入力します。

```
%U
```

メッセージテキストを追加するときに入力するすべての小文字は、現在のNaturalセッション中に大文字に変換されます。

%U の詳細については、『*端末コマンド*』ドキュメントを参照してください。

2 SYSERR メインメニューを呼び出し、次の値を入力します。

フィールド	入力値
Code	AD
Message type	NS Natural システムショートメッセージ NL Natural システムロングメッセージ US ユーザー定義ショートメッセージ UL ユーザー定義ロングメッセージ ロングメッセージは、ショートメッセージの説明であるため、対応するショートメッセージがある場合にのみ追加できます。
Library	任意の既存 Natural ライブラリ。
Message number	追加するメッセージの範囲を示す、最初と最後のメッセージの番号（最大4桁）。メッセージを 1 つのみ追加する場合は、左側の［**Message number**］フィールドに新しいメッセージの番号を入力して右側のフィールドをクリアするか、または両方のフィールドに番号を入力します。
Language codes	メッセージを追加する言語のコード。メッセージタイプが NS または NL の場合、言語コードは英語の 1 である必要があります。その他のメッセージタイプの場合は、フィールドに入力した最初の言語コードが使用され、他の言語は無視されます。

3 ENTER キーを押します。

次のような［**Add Short Message**］画面が表示されます。

```
15:53:03              ***** NATURAL SYSERR UTILITY *****            2008-11-28
                            - Add Short Message -

Number        Short Message (Language code=1)
------------  ---------------------------------------------------------------
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
```

［**Number**］フィールドには、メッセージ番号（上記の例では1004）が表示されます。メッセージ番号の前にはライブラリ ID が付きます（上記の例では SYSERR）。

4 メッセージ番号の横の入力行にショートメッセージテキストを入力し、ENTER キーを押します。

または:

［**Sample**］というラベルの付いた行に上記の例に示すようなサンプルメッセージテキストが含まれている場合は、.C を入力して ENTER キーを押し、このテキストを入力行にコピーします。サンプルメッセージテキストに文字列 0000 が含まれている場合、次の例で示すように、この文字列は新しいメッセージ番号に置き換えられます。

```
15:57:14              ***** NATURAL SYSERR UTILITY *****            2008-11-28
                                - Add Short Message -

Number        Short Message (Language code=1)
------------  ---------------------------------------------------------------
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  ↩
 ↩
```

サンプルメッセージの作成方法については、「*ダイレクトコマンド*」の SAMPLE コマンドの説明を参照してください。

5 対応するロングメッセージテキストを追加するには、PF9 キーを押します。

次のような［**Add Long Message**］画面が表示されます。

```
11:21:59          - Add Long Message SYSERR1004 Language 1 -         2003-09-16
 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                    -     +           Copy        Canc
```

6 3つの入力エリア［**Tx.**］（テキスト）、［**Ex.**］（説明）、［**Ac.**］（アクション）にテキストを入力します。

7 ENTER キーを押して、ロングメッセージを保存します。

8 PF9 キーを押して、ショートメッセージに戻るか、またはメッセージ番号の範囲を選択している場合は次のショートメッセージを昇順で追加します。

9 PF3 キーまたは PF12 キーを押して、SYSERR メインメニューに戻ります。

または:

メッセージ番号の範囲を選択している場合は、PF8 キーまたは PF7 キーを押して、次のショートメッセージを昇順または降順で追加します。

メッセージの削除

手順 46.2. メッセージを削除するには

■ SYSERR メインメニューのフィールドに、次の値を入力します。

フィールド	入力値
Code	DE
Message type	NS Natural システムショートメッセージ NL Natural システムロングメッセージ US ユーザー定義ショートメッセージ UL ユーザー定義ロングメッセージ 対応するショートメッセージを削除しないでロングメッセージを削除することはできますが、対応するロングメッセージを削除しないでショートメッセージを削除することはできません。ロングメッセージが存在するショートメッセージを削除すると、両方削除することを確認するメッセージが表示されます。
Library	任意の既存 Natural ライブラリ。
Message number	削除するメッセージの範囲を示す、最初と最後のメッセージの番号（最大4桁）。
Language codes	メッセージを削除する言語のコード。指定したメッセージが、使用可能なすべての言語で削除されることを示すには、アスタリスク（*）を入力します。

メッセージの表示

手順 46.3. メッセージを表示するには

1 SYSERR メインメニューのフィールドに、次の値を入力します。

フィールド	入力値
Code	DI
Message type	NS Natural システムショートメッセージ NL Natural システムロングメッセージ US ユーザー定義ショートメッセージ UL ユーザー定義ロングメッセージ
Library	任意の既存 Natural ライブラリ。
Message number	表示するメッセージの範囲を示す、最初と最後のメッセージの番号（最大4桁）。
Language codes	メッセージを表示する言語のコード。指定可能な言語コードは 1 つのみです。複数の言語コードを指定した場合は、最初のコードのみが使用され、他のコードはすべて無視されます。

2 ENTER キーを押します。

ショートメッセージの場合、次のような［**Display Short Messages**］画面が表示されます。

```
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
```

次のページを表示するには、PF8 キーを押します。

ロングメッセージの場合、［**Display Long Message**］画面が表示されます。PF8 キーを押して次のページを表示するか、または PF7 キーを押して前のページを表示すると、メッセージが 1 つずつ表示されます。［**Display Long Message**］画面は、「*メッセージの変更*」で説明する［**Modify Long Message**］画面に類似しています。

メッセージの変更

手順 46.4. メッセージを変更するには

1 SYSERR ユーティリティでは、デフォルトで大文字と小文字が区別されます。変更されるメッセージで大文字と小文字の変換を行う場合は、次の端末コマンドを入力します。

```
%U
```

メッセージテキストを編集するときに入力するすべての小文字は、現在の Natural セッション中に大文字に変換されます。

%U の詳細については、『*端末コマンド*』ドキュメントを参照してください。

2 SYSERR メインメニューのフィールドに、次の値を入力します。

フィールド	入力値
Code	MO
Message type	NS Natural システムショートメッセージ NL Natural システムロングメッセージ US ユーザー定義ショートメッセージ UL ユーザー定義ロングメッセージ
Library	任意の既存 Natural ライブラリ。
Message number	変更するメッセージの範囲を示す、最初と最後のメッセージの番号（最大4桁）。
Language codes	メッセージを変更する言語のコード。指定可能な言語コードは 1 つのみです。複数の言語コードを指定した場合は、最初のコードのみが使用され、他のコードはすべて無視されます。

3 ENTER キーを押します。

次のような［**Modify Short Message**］画面が表示されます。

```
18:52:33              ***** NATURAL SYSERR UTILITY *****             2003-09-16
                          - 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
```

参照用に、ロングメッセージが画面の下半分に表示されます。

ロングメッセージを変更する場合は、［**Modify Long Message**］画面が表示されます。

```
18:54:02      - Modify Long Message SYSERR1004 (English) -          2003-09-16
 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 ENTER キーを押して変更を保存します。

5 メッセージ番号の範囲を選択している場合は、PF8 キーまたは PF7 キーを押して、次のメッセージを昇順または降順で変更します。

メッセージの出力

手順 46.5. メッセージを出力するには

1 SYSERR メインメニューのフィールドに、次の値を入力します。

フィールド	入力値
Code	PR
Message type	NS Natural システムショートメッセージ NL Natural システムロングメッセージ US ユーザー定義ショートメッセージ UL ユーザー定義ロングメッセージ

フィールド	入力値
Library	任意の既存 Natural ライブラリ。
Message number	出力するメッセージの範囲を示す、最初と最後のメッセージの番号（最大4桁）。
Language codes	メッセージを出力する言語のコード。指定可能な言語コードは 1 つのみです。複数の言語コードを指定した場合は、最初のコードのみが使用され、他のコードはすべて無視されます。

2 ENTER キーを押します。

以下の例のようなプリントウィンドウが表示されます。

```
+------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 プリントウィンドウに表示されているオプションおよび論理プリンタ名を指定します。

論理プリンタ名の詳細については、『*Natural ステートメント*』のステートメント「DEFINE PRINTER」を参照してください。

4 ENTER キーを押して、選択したメッセージをプリンタに出力します。

手順 46.6. すべての Natural システムメッセージを印刷するには

■ SYSERR メインメニューのフィールドに、次の値を入力します。

- コード PR、
- メッセージタイプ NS または NL、
- メッセージ番号の範囲 1～9999,
- 言語コード 1（英語）または 2（ドイツ語）

ライブラリ ID は必要なく、入力しても無視されます。

メッセージのスキャン

この機能を使用すると、メッセージをスキャンして特定の文字列を検索できます。スキャンできるのは、ショートメッセージのみです。

手順 46.7. メッセージをスキャンするには

1 SYSERR メインメニューのフィールドに、次の値を入力します。

フィールド	入力値
Code	SC
Message type	NS Natural システムショートメッセージ US ユーザー定義ショートメッセージ
Library	任意の既存 Natural ライブラリ。
Message number	スキャンするメッセージの範囲を示す、最初と最後のメッセージの番号（最大 4 桁）。
Language codes	1～9、A～Z、および a～y を使用して 9 文字以内の言語コードを指定するか、またはアスタリスク（*）を入力してすべての言語を指定します。

2 ENTER キーを押します。

以下の例のようなスキャンウィンドウが表示されます。

```
+-------------------------------------+
|   Scan value(s)       Or/And/Not    |
|   ----------------    ----------    |
|   ________________       OR_        |
|   ________________                  |
|   ________________                  |
|   ________________                  |
|                                     |
|   Absolute ..... X                  |
|   Immediate .... _                  |
|                                     |
+-------------------------------------+
```

表示されるフィールドで、スキャンに使用する検索条件を指定できます。

Scan value(s)	4 つの空のフィールドに、検索する文字列を 4 つまで入力します。スキャンでは、大文字および小文字の該当する用語が検索されます。
Or/And/Not	次のいずれかの演算子を入力して、ブール検索クエリを実行できます。 `OR` ［**Scan value(s)**］に入力された文字列を 1 つまたは複数検索します。これはデフォルト設定です。 `AND` ［**Scan value(s)**］に入力された文字列のすべてを検索します。 `NOT` ［**Scan value(s)**］に入力された文字列をいずれも検索しません。 1 つの［**Scan value(s)**］フィールドにのみ入力した場合、演算子は無視されます。
Absolute	このフィールドをマークすると、単語に含まれている文字列も検索されます。例えば、値「`meter`」を探してスキャンすると、「parameter」や「millimeter」などの単語も検索されます。 マークを削除すると、検索は単語全体にのみ一致するように制限されます。
Immediate	このフィールドをマークすると、個別のメッセージが順に表示されます。マークしない場合は、検索の終了後にメッセージのリストが表示されます。 ［**Language codes**］フィールドで複数の言語またはアスタリスク（*）を指定した場合は、［**Immediate**］をマークする必要があります。

3 次の例に示すように、検索条件を指定します。

```
+-------------------------------------+
!   Scan value(s)       Or/And/Not    !
!   ----------------    ----------    !
!   BUFFER___________       AND       !
!   POOL_____________                 !
!   _________________                 !
!   _________________                 !
!                                     !
!   Absolute ..... X                  !
!   Immediate .... _                  !
!                                     !
+-------------------------------------+
```

上記の例では、次の両方を含むすべてのショートメッセージが検索されます：`buffer` と `pool`.

4 ENTER キーを押します。

次の例に示すように、指定した検索条件が適用されるすべてのメッセージが画面に表示されます。

```
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
```

検索文字列が含まれる単語は強調表示されます。

この画面でPF10キーを押すと、現在のスキャンで使用されている検索条件を表示できます。

リストからのメッセージの選択

この機能を使用すると、メッセージの範囲を表示し、さらに処理する個別のメッセージを選択できます。表示できるのは、ショートメッセージのみです。

手順 46.8. メッセージを選択するには

1 SYSERR メインメニューのフィールドに、次の値を入力します。

フィールド	入力値
Code	SE
Message type	NS Natural システムショートメッセージ US ユーザー定義ショートメッセージ

フィールド	入力値
Library	任意の既存 Natural ライブラリ。 ライブラリ ID にアスタリスク（*）が追加されている場合は、使用可能なすべてのライブラリのリストが選択用に表示されます。
Message number	選択用に表示するメッセージの範囲を示す、最初と最後のメッセージの番号（最大 4 桁）。
Language codes	メッセージを表示する言語のコード。 複数の言語コードを指定すると、最初のコードのショートメッセージのみが表示されます。各メッセージの使用可能な言語を表示するには、アスタリスク（*）を入力します。

2 ENTER キーを押します。

次のような［**Select Messages**］画面が表示されます。

```
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
__
__
__
```

3 ［**Se**］列で、次の行コマンドのいずれかを入力します。

Command	説明
DE	メッセージを削除します。
DI	［**Select Messages**］画面を、「*メッセージの表示*」で説明する［**Display Short Messages**］画面に変換します。また、このコマンドを使用して選択したメッセージをリストの先頭に配置し、「.X」で説明されているように表示されるメッセージの数を減らします。
LA	翻訳が存在する言語のコードを表示します。
MO	メッセージを変更します。
PR	メッセージをプリンタに出力します。
SH	ショートメッセージを表示します。 このコマンドは、SYSERR メインメニューの［**Language codes**］フィールドにアスタリスク（*）が入力されている場合にのみ使用できます。
TR	メッセージを別の言語に翻訳します。
.X	選択したメッセージをリストの先頭に配置して、より狭いメッセージ範囲を定義します。これにより、表示されるメッセージの数が削減されます。 このコマンドを使用して選択したメッセージがリストの先頭に配置され、このメッセージの上にリストされていたメッセージは表示から削除されます。これにより、SYSERR メインメニューのメッセージ範囲がリセットされ、［**Select Messages**］画面はここで選択したメッセージで始まります。
.Y	選択したメッセージ以前のメッセージのみをリストして、より狭いメッセージ範囲を定義します。 このコマンドを使用して選択したメッセージ以降にリストされていたすべてのメッセージは、表示から削除されます。これにより、SYSERR メインメニューのメッセージ範囲がリセットされ、［**Select Messages**］画面はここで選択したメッセージで終わります。

4 ENTER キーを押して次に進みます。

メッセージの他の言語への翻訳

この機能を使用すると、ある言語のショートメッセージを1つ以上の他の言語に翻訳できます。ロングメッセージを他の言語に翻訳するには、「*メッセージの追加*」の手順に従います。

手順 46.9. ショートメッセージを翻訳するには

1 SYSERR メインメニューのフィールドに、次の値を入力します。

フィールド	入力値
Code	TR
Message type	NS Natural システムショートメッセージ US ユーザー定義ショートメッセージ
Library	任意の既存 Natural ライブラリ。
Message number	選択用に表示するメッセージの範囲を示す、最初と最後のメッセージの番号（最大 4 桁）。
Language codes	言語コードを 9 つまで指定します。言語コードは、1〜9、A〜Z、および a〜y の単一英数字です。

2 ENTER キーを押します。

次のような［**Translate Short Message**］画面が表示されます。

```
13:42:31             ***** NATURAL SYSERR UTILITY *****              2009-01-16
                         - 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
```

［**Languages**］フィールドには、メッセージがすでに存在する言語コードが表示されます（上の例では、1、4、および 5）。

以下のセクションでは、［**Number**］および［**Languages**］に、SYSERR メインメニューの［**Language codes**］フィールドに事前に言語コードが入力されたすべての言語を示します（上記の例では1、2、3、4、5）。英語（1）、スペイン語（4）、イタリア語（5）の翻訳はすでに存在しますが、ドイツ語（2）とフランス語（3）の翻訳は新しく入力できます。

参照用に、画面下部には、言語／ショートメッセージのセクションで最初にリストされている言語（画面例では英語）に対応しているロングメッセージが3行表示されます。行1、4、および 18 は、デフォルトで表示されます。ロングメッセージの他の行を表示するには、3 つの行番号（**1**、**4**、または **18**）を別の行番号で上書きし、ENTER キーを押します。

3 指定した新しい言語の横の入力行に翻訳を入力します。

4 ENTER キーを押します。

手順 46.10. ショートメッセージの翻訳を変更するには

1 ［**Translate Short Message**］画面で PF10 キーを押します。

以下の例のような［**Options**］ウィンドウが開きます。

```
+------------------------ Options ------------------------+
!                                                          !
!  Modification of all fields allowed ....... N            !
!                                                          !
!  Currently recognized language codes ...... 123456789    !
!                                                          !
+----------------------------------------------------------+
```

2 上のフィールドで N（デフォルト）を Y に置き換え、下のフィールドで必要な言語コードを入力します。翻訳する新しい言語コードを 9 個まで指定できます。

手順 46.11. 翻訳を空の入力行にコピーするには

1 ［**Translate Short Message**］画面で、空の行の最初の 2 つの位置に .C を入力します。

2 別の言語ですでに存在しているショートメッセージの任意の場所にカーソルを合わせます（コピーできるのは表示モードで表示されているテキストのみです）。

3 ENTER キーを押します。

47 パラメータ

- Message Type 298
- ライブラリ 298
- メッセージ番号 298
- Language Codes 298

このセクションでは、SYSERR メインメニューの機能に指定できるパラメータについて説明します。特定の機能でのパラメータの使用に適用される制限については、「*機能*」で説明しています。

Message Type

処理するメッセージのタイプを指定します。次のテーブルに、使用可能なメッセージタイプを示します。

Type	説明
NS	Natural システムショートメッセージ
NL	Natural システムロングメッセージ
US	ユーザー定義ショートメッセージ
UL	ユーザー定義ロングメッセージ

ライブラリ

メッセージが作成されるか、または管理されるライブラリを指定します。Natural システムメッセージ（メッセージタイプ NS および NL）にアクセスする場合は、ライブラリを指定する必要がありません。［**Library**］フィールドの入力値は無視されます。

メッセージ番号

メッセージ範囲の最初の番号と最後の番号を指定します。ライブラリおよび言語の最大メッセージ番号は 9999 です。メッセージ番号 0000 は使用できません。メッセージ番号を 1 つのみ指定するには、左側の［**Message number**］フィールドにメッセージ番号を入力するか、または両方のフィールドに番号を入力します。

Language Codes

使用可能な言語コードを 9～60 個指定します。言語コードは、1～9、A～Z、および a～y の単一英数字です。言語コードを表示または選択するには、［**Language codes**］フィールドの先頭に疑問符（?）を入力して、ENTER キーを押します。詳細については、『システム変数』ドキュメントのシステム変数「*LANGUAGE」を参照してください。

48 ダイレクトコマンド

SYSERR メインメニューでは、次のコマンドをコマンド行に入力して実行できます。

Command	説明	
EXPORT	メッセージファイルをエクスポートし、テキストファイルに変換します。必ずファイルの完全パスを指定してください。 ファイルフォーマットとメッセージファイルおよびテキストファイルの生成の詳細については、関連するセクションを参照してください。	
	From (application)	テキストファイルの生成元となるメッセージファイル／アプリケーション（ライブラリ）。 ユーザー定義メッセージ：エクスポートウィンドウに入力されているデフォルト値は、SYSERR メインメニューの［**Message type**］フィールド、［**Library**］フィールド、および［**Language codes**］フィールドに入力された値から取得されています。生成されたメッセージファイルには、指定した範囲に関係なく、すべてのメッセージが含まれます。 Natural システムメッセージ：デフォルトのディレクトリパス（必要に応じて）およびファイル名を変更します。ファイル名は、N*nn*L*mmmm*.MSG にする必要があります。
	To (text file)	自動的に生成されるテキストファイルの名前。

Command	説明
IMPORT	テキストファイルをインポートし、メッセージファイルに変換します。必ずファイルの完全パスを指定してください。 ファイルフォーマットとメッセージファイルおよびテキストファイルの生成の詳細については、関連するセクションを参照してください。 **From (text file)**：メッセージファイルの生成元となるテキストファイルの名前。 **To (application)**：テキストファイルが生成されるメッセージファイル／アプリケーション（ライブラリ）。 ユーザー定義メッセージ：インポートウィンドウに入力されているデフォルト値は、SYSERR メインメニューの［**Message type**］フィールド、［**Library**］フィールド、および［**Language codes**］フィールドに入力された値から取得されています。生成されたメッセージファイルには、指定した範囲に関係なく、すべてのメッセージが含まれます。 Natural システムメッセージ：ファイル名は、N*nn*L*mmmm*.MSG にする必要があります。
LAYOUT	有効なメッセージ範囲を指定して、メッセージを分類します。範囲を重複させることができます。新しいメッセージを追加できるのは、その番号がレイアウトで指定した範囲内にある場合のみです。
NEXT	指定されているメッセージ番号範囲内で、次のフリーメッセージ番号を検索します。フリーとは、このメッセージ番号が使用可能で、どの言語のメッセージファイルにも割り当てられていないことを意味します。
NEXTTAB	NEXT と同じですが、番号を選択するためのメッセージ番号のリストを返します。
RESTART	ユーティリティを終了しないで、SYSERR（およびそのデフォルト値）を再初期化します。

Command	説明
SAMPLE	［**Edit SAMPLE message**］ウィンドウを呼び出します。このウィンドウでは、新しいショートメッセージの作成時にマスタとして使用するサンプルメッセージを作成したり、変更したりします。 サンプルメッセージを作成または変更するには、次の手順に従います。 ■ ［**Edit SAMPLE message**］ウィンドウのエディタエリアで、必要なメッセージテキストを入力するか、または既存のテキストを変更します。文字列 0000（テキストと組み合わせてまたは組み合わせずに）を入力すると、この文字列0000は、新しいメッセージのコピー時にそのメッセージの番号で置き換えられます。「*機能*」セクションの「*メッセージの追加*」の手順 **4** も参照してください。 ■ ［**Read or Write sample**］フィールドに W を入力して、エントリを保存します。 ■ ［**Library**］フィールドに、サンプルメッセージを使用するライブラリの名前を入力します。［**Library**］フィールドに入力しない場合、サンプルメッセージは Natural システムメッセージに適用されます。 ■ PF3 キーを押して、［**Edit SAMPLE message**］ウィンドウを閉じます。 各言語およびライブラリにサンプルメッセージを 1 つ定義できます。
SHIFT	有効にすると、新しいメッセージの変更または追加を確定したときに、ショートメッセージのテキストが左側の余白に自動的にシフトされます。
TRACE	データベースアクセス数をカウントします。指定したメッセージ番号に達すると、ウィンドウが表示されます。デフォルトの番号は 900 です。0 に設定すると、トレース機能が終了します。コマンド TRACE ON および TRACE OFF は、コマンド行に直接入力できます。TRACE ON により、アクセスカウンタが 900 に設定され、TRACE OFF により、アクセスカウンタが 0 に設定されます。
USEREXIT	Natural システムライブラリ SYSERR で USEREXIT プログラムを呼び出します。

49 大文字変換 - ERRUPPER

Naturalシステムメッセージは小文字で表示されます。使用している端末で小文字が適切に表示されない場合は、Natural システムライブラリ SYSERR でプログラム ERRUPPER を実行して、メッセージを小文字から大文字に変換します。

ただし、大文字に変換したメッセージを再度小文字に変換することはできません。小文字メッセージを復元するには、次の 2 つの方法があります。

- オブジェクトハンドラを使用して、メッセージを再ロードします。
- 変換前にオブジェクトハンドラを使用して、フリー言語コードへの小文字メッセージをアンロードします。これにより、常にバックアップが存在します。

詳細については、『*オブジェクトハンドラ*』ドキュメントを参照してください。

50 文字置換 - ERRCHAR

使用している端末で特定の文字が適切に表示されない場合は、その文字を検索して、任意の新しい文字で置換できます。これには、Naturalシステムライブラリ SYSERR でプログラム ERRCHAR を実行します。ただし、置換できるのは Natural システムショートメッセージ内の文字のみです。ERRCHAR を使用して、特定の文字をスキャンし、この文字を表す 16 進コードを別の 16 進コードで置き換えます。

プログラム ERRCHAR を実行すると、次の機能を持つ［**ERRCHAR**］メニューが表示されます。

- 指定された文字のスキャン
- 文字のスキャンと置換
- メッセージの 16 進形式での表示
- 使用している端末用の ASCII 文字テーブル
- 文字セット ERRCSET を使用した翻訳

［**ERRCHAR**］メニューには、次の入力フィールドがあります。

フィールド	説明
Message Number	検索または検索／置換操作に含まれるメッセージの範囲。
Language Code	検索または検索／置換操作に含まれる Natural システムショートメッセージの言語コード。
Scan Value	スキャン対象となる 16 進値。
Replace Value	検索されたすべてのスキャン対象値を置き換える 16 進値。［**ASCII character table for your terminal**］機能を使用すると、使用している端末で表示できる文字を調べることができます。

51 メッセージファイルとテキストファイルの生成

- メッセージファイルの保存 308
- テキストファイルの作成 308
- メッセージファイルの生成 309
- テキストファイルの再作成 310

Natural以外の任意の環境でメッセージをテキストファイルとして作成し、メッセージファイルに変換して、SYSERR ユーティリティで管理できます。メッセージファイルは、SYSERR ユーティリティのインポート機能およびエクスポート機能によって作成され、管理されます。

メッセージファイルはプラットフォームに依存しないフォーマットで作成されます。このフォーマットは、Natural でサポートされている UNIX、OpenVMS、および Windows プラットフォーム間で移植できます。例えば、Natural for Windows 環境で作成されたメッセージファイルは、手動で変換することなく UNIX プラットフォームまたは OpenVMS プラットフォームにコピーできます。必要なエンディアン変換は、Naturalが行います。詳細については、『*オペレーション*』ドキュメントの「*Natural の移植可能なシステムファイル*」および『*プログラミングガイド*』の「*Natural 生成プログラムの転送*」を参照してください。

メッセージファイルの保存

メッセージファイルはファイル拡張子 .MSG を付けて、Natural Err ディレクトリに保存する必要があります。

ユーザー定義メッセージファイルは、アプリケーションの実行元となる FNAT システムファイルまたは FUSER システムファイルのライブラリ、STEPLIB、または SYSTEM ライブラリの Err サブディレクトリに保存されます。

Natural システムメッセージの場合、メッセージファイルは Natural ルートディレクトリの Err サブディレクトリに保存する必要があります。Natural システムメッセージは、8 つのメッセージファイルに保存されます。

テキストファイルの作成

Natural システムメッセージまたはユーザー定義メッセージの場合、SYSERR ユーティリティのインポート機能によって 1 つのテキストファイルからメッセージファイルが生成されます。

このようなテキストファイルを作成する場合は、次の例に示すようなレイアウトを使用する必要があります。

例：

```
NAT
0010
0100
0010E NO MESSAGE TEXT DEFINED!
0020E MISSING/INVALID SYNTAX; UNDEFINED VARIABLE-NAME.
0025E ERROR IN ENTRY FOR NUMBER OF RECORDS TO BE PROCESSED.
0050E INCORRECT FIELD SPECIFICATION IN 'WHERE' CLAUSE.
#PLEASE CHECK PROGRAM
```

```
#FOR ERRORS
0100E FUNCTION NOT AVAILABLE.
```

説明

NAT または *library-ID*	メッセージとともに表示されるメッセージ番号の接頭辞。デフォルトの接頭辞は、Natural システムメッセージの場合は NAT で、ユーザー定義メッセージの場合はライブラリ ID です。
0010	メッセージ範囲の 4 桁の開始番号
0100	メッセージ範囲の4桁の終了番号このテキストファイルで定義されているすべてのメッセージ番号は、この範囲内である必要があります。
0010E	NO MESSAGE TEXT DEFINED! メッセージ番号 0010 のショートメッセージ。E は必須で、エラー（error）を意味します。このメッセージは、次の Natural ステートメントとともに発行されます。 REINPUT *0010 説明のためのロングメッセージは、このショートメッセージのすぐ下に配置する必要があります。ロングメッセージ用に追加する各行は、先頭にハッシュ記号／番号記号（#）を付ける必要があります。各ショートメッセージに対して、20 行以下のロングメッセージテキストを追加できます。

メッセージファイルの生成

SYSERR ユーティリティには、テキストファイルからメッセージファイルを生成するオプションがあります。

ユーザー定義メッセージの場合、各ライブラリの1つの言語について出力メッセージファイルを1つ作成できます。各メッセージファイルは、ライブラリの Err サブディレクトリに保存する必要があります。

命名規則

ユーザー定義メッセージの場合は、次のメッセージファイル名を使用する必要があります。

```
NnnAPMSL.MSG
```

nn は言語コード（01〜60）です。例えば、英語の場合は 01 です。

Naturalシステムメッセージの場合は、次のメッセージファイル名を使用する必要があります。

```
NnnLmmmm.MSG
```

nn は使用される言語コードで、*mmmm* はメッセージ範囲の開始番号です。メッセージ番号の範囲は、Naturalシステムのインストール時に定義したとおりに固定されています。例えば、次のようになります。

N01L0000	メッセージ 1〜1999
N01L2000	メッセージ 2000〜2999

手順 51.1. メッセージファイルを生成するには

1 SYSERR ユーティリティの IMPORT コマンドを入力します。

［**Import Text File to Message File**］ウィンドウが表示されます。

2 ［**From**］入力フィールドで、すべての情報の読み込み元となる入力テキストファイルの名前を指定します。完全パス名を指定する必要があります。［**To**］入力フィールドで、生成されるメッセージファイルの言語およびライブラリを指定します。

テキストファイルの再作成

SYSERRユーティリティには、メッセージテキストの管理用にテキストファイルを再作成するオプションがあります。再作成するには、メッセージファイルをテキストファイルに再変換します。

手順 51.2. メッセージテキストファイルを再作成するには

1 SYSERR ユーティリティの EXPORT コマンドを入力します。

［**Export Text File from Message File**］ウィンドウが表示されます。

2 ［**From**］入力フィールドで、入力として使用されるメッセージの言語およびライブラリを指定します。［**To**］入力フィールドで、作成するテキストファイルの名前を指定します。作成されるテキストファイルは、入力テキストファイルと同じフォーマットになります。

52 さまざまなライブラリのメッセージの管理

「*ダイレクトコマンド*」で説明されているように EXPORT コマンドおよび IMPORT コマンドを使用するか、関連ドキュメントで説明されているように*オブジェクトハンドラ*を使用して、メッセージをライブラリ間で転送できます。

また、オブジェクトハンドラを使用して、別のライブラリ内のメッセージの移動、名前変更、検索、リスト、または削除を行うこともできます。

53 アプリケーションプログラミングインターフェイス USR0020P

Natural システムライブラリ SYSEXT のアプリケーションプログラミングインターフェイス USR0020P を使用すると、FNAT システムファイルまたは FUSER システムファイルからメッセージを読み取ることができます。これにより、例えば、Natural システムライブラリ SYSERR を使用せずにロングメッセージをアプリケーションに表示できます（独自のユーザー定義ヘルプシステムとして）。

Natural システムライブラリ SYSEXT にログオンし、コマンド行にコマンド `MENU` を入力します。表示されるリストで、疑問符（?）を使用してプログラム USR0020P をマークします。プログラムで実行する機能を選択できるウィンドウが表示されます。`I` を入力すると、USR0020P の使用についての詳細情報が表示されます。

VII SYSEXT ユーティリティ - Natural アプリケーションプログラミングインターフェイス

54 SYSEXT ユーティリティ - Natural アプリケーションプログラミングインターフェイス

- SYSEXT の概要 318
- SYSEXT の呼び出しと終了 319
- SYSEXT ユーティリティの使用 321
- インターフェイスバージョン 326
- 予約キーワード 326
- Natural API の使用 327

SYSEXT ユーティリティを使用すると、現在のシステムライブラリ SYSEXT に含まれている Natural アプリケーションプログラミングインターフェイス（API）を検索し、テストできます。

Natural API は Natural サブプログラム（カタログ化オブジェクト）で、データにアクセスして必要に応じて変更したり、Natural ステートメントではアクセスできないサービスを実行したりする場合に使用します。Natural API は、Natural、サブコンポーネント、またはサブ製品を参照します。

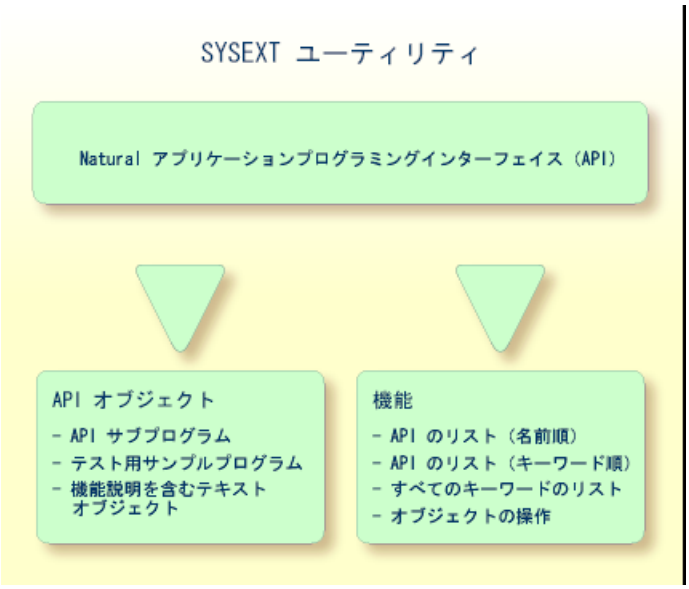
関連トピック：

- 「*Application Programming Interfaces*」 - 『*Natural Security*』ドキュメント

SYSEXT の概要

SYSEXT ユーティリティには、各 Natural API について、機能説明、1 つのプログラム例、1 つのカテゴリ、および API に固有のキーワードがあります。

次の図に、SYSEXT で提供されている Natural オブジェクトおよび主な機能の概要を示します。



Natural API に提供されているオブジェクト

各 Natural API に提供されている一般的な Natural オブジェクトのタイプは、次のセクションでリストされています。特定の API 用に存在する可能性のある追加オブジェクトは含まれていません。

API のすべての関連オブジェクトは、システムファイル FNAT のライブラリ SYSEXT に含まれています。

次の表の *nnnn* は、API、および対応するプログラム例とテキストオブジェクトを識別するための 4 桁の番号を表します。

Object Name	説明
USR*nnnn*N	指定された機能を実行する API サブプログラム（カタログ化オブジェクト）。
USR*nnnn*P	API の効果のテストに使用できるプログラム例（ソースオブジェクト）。 プログラム例によって、対応するサブプログラム USR*nnnn*N が呼び出されます。
USR*nnnn*T	短い説明と長い説明をリスト表示するテキストオブジェクト、および使用法、キーワード、カテゴリ、インターフェイスバージョンに関する情報。 テキストオブジェクトを表示するには、行コマンド T を使用します。「*行コマンド*」を参照してください。

一部の API では、API に関連する機能を提供するコピーコードを使用できます。コピーコードには USR*nnnnX* という名前が付けられています。*X* は識別文字（"Z"、"Y" など）です。

SYSEXT の呼び出しと終了

SYSEXT ユーティリティは、次の構文図で説明されているシステムコマンド SYSEXT によって起動されます。

```
SYSEXT [ ALL     ] [ FIRST  ] [ DESCENDING ]
       [ CURRENT ] [ SECOND ] [ ASCENDING  ]
```

次の表に、パラメータの説明を示します。

パラメータ	説明
ALL	すべての API をリストします（デフォルト）。
CURRENT	現在の API、つまりすべてのユニークな API、およびインターフェイスバージョンを持つ API の現在のバージョン（キーワード +CURRENT-VERSION を使用）のみをリストします。「*インターフェイスバージョン*」を参照してください。
FIRST	最初のメニューと製品コード項目（デフォルト）を表示します。以下の例を参照してください。

パラメータ	説明
SECOND	2 番目のメニューとカテゴリ項目を表示します。以下の例を参照してください。
ASCENDING	API を昇順で表示します（デフォルト）。
DESCENDING	API を降順で表示します。

手順 54.1. SYSEXT を呼び出すには

■ 次のシステムコマンドを入力します。

```
SYSEXT
```

以下の例に示すようなメニューが表示されます。最初のメニューに応じて、使用可能なすべての Natural API のリストが表示されます。

```
10:57:31              ***** NATURAL SYSEXT UTILITY *****             2012-04-23
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
 _  USR0080N  Get or set type and name of editor contents                  NAT
 _  USR0120N  Read Natural short error message                              NAT
 _  USR0210N  Save, catalog or stow Natural object                          NAT
 _  USR0220N  Read Natural long error message                               NAT
 _  USR0221N  Read Natural long error message                               NAT
 _  USR0320N  Read user short error message from FNAT or FUSER              NAT
 _  USR0330N  Read Natural object directory                                 NAT

 Category .. ________________________________  Keyword .. ____________________

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

任意のパラメータで SYSEXT を呼び出すことができます。例えば、起動時に 2 番目のメニューを表示することができます。

```
SYSEXT SECOND
```

この場合は、次のようなメニューが表示されます。

```
10:57:31              ***** NATURAL SYSEXT UTILITY *****            2012-04-23
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
 _  USR0080N  Get or set type and name of editor   EDITOR
 _  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
 _  USR0221N  Read Natural long error message      ERROR MESSAGES
 _  USR0320N  Read user short error message from   ERROR MESSAGES
 _  USR0330N  Read Natural object directory        NATURAL OBJECTS

 Category .. ________________________________   Keyword .. ____________________
Command ===>
Enter-PF1---PF2---PF3---PF4---PF5---PF6---PF7---PF8---PF9---PF10--PF11--PF12---
      Help  Reset Exit  Desc  Curr  --    -     +     ++    <           Canc
```

別の方法として、パラメータなしで SYSEXT を呼び出し、表示されるメニューを調整することもできます。「*PF キー*」を参照してください。

手順 54.2. SYSEXT を終了するには

■ SYSEXT ユーティリティメニューで PF3 キーまたは PF12 キーを押します。

または:

コマンド行にピリオド（.）または EXIT を入力します。

SYSEXT ユーティリティの使用

このセクションでは、次のトピックについて説明します。

- SYSEXT ユーティリティメニューの要素
- PF キー
- 行コマンド

- ユーティリティコマンド

SYSEXT ユーティリティメニューの要素

SYSEXT ユーティリティメニューには、API のリストが表示されます。各 API には、API 名（［**Interface**］）、説明（［**Description**］）、および選択したメニューに応じて製品コード（［**Prod**］）またはカテゴリ（［**Category**］）による分類があります。左端の列（［**Cmd**］）を除き、各列のヘッダーは選択条件を入力できる選択フィールドです。

メニューの下部にある［**Category**］検索フィールドを使用して、使用可能なカテゴリを検索することもできます。

［**Keywords**］入力フィールドは、使用可能なキーワードを取得するための検索フィールドとして使用することも、API を取得するための選択フィールドとして使用することもできます。

複数選択フィールドで選択条件を指定することもできます。例えば、キーワード `PF-KEY` およびカテゴリ `NATURAL ENVIRONMENT` を使用して、`USR4` で始まる API を 1 つの手順で選択できます。

すべてのメニュー要素とその使用方法の詳細を以下に示します。

要素	説明	使用方法
Cmd	テキストオブジェクトまたはプログラム例で実行される行コマンドの入力フィールド。「*行コマンド*」を参照してください。	行コマンドを入力します。例： `K` 指定されている API に関連するすべてのキーワードをリストします。
インターフェイス	API サブプログラムの名前。インターフェイスバージョンを持つ API が強調表示されます。	アスタリスク（`*`）を入力するか、アスタリスクで区切った接頭辞を入力します。例： `USR4*` 接頭辞 `USR4` を持つすべての API をリストします。
説明	API の目的の概要。	文字列を入力します。例： `default` `default` または `Default` という文字列が説明に含まれているすべての API をリストします。 `*` `*` という文字列が説明に含まれているすべての API をリストします。
Prod	API の影響を受ける Natural（`NAT`）または Natural アドオン製品の製品コード。 使用可能な製品コード：`NAT` = Natural、, `NDV` = Natural 開発サーバー、`PRD` = Predict、`RPC` = Natural RPC（リモートプロシージャコール）	名前を入力するか、アスタリスク（`*`）で区切った接頭辞を入力します。例： `N*` 製品コードが `N` で始まるすべての API をリストします。

要素	説明	使用方法
	NAT 以外の製品コードは、強調表示されます。 最初のメニューにのみ表示されます。	
カテゴリ（選択フィールド）	API が機能エリアまたは目的に従って分類されるカテゴリ。 1 つの API には 1 つのカテゴリしか設定できません。 2 番目のメニューにのみ表示され、API のリストの上に配置されます。	名前を入力するか、アスタリスク（*）で区切った接頭辞を入力します。例： NATURAL OBJECTS カテゴリが NATURAL OBJECTS であるすべての API をリストします。
Category（検索フィールド）	すべてのカテゴリをリストします。 API のリストの下にあります。	「*カテゴリ検索*」を参照してください
キーワード	すべてのキーワードをリストします。または選択条件としてキーワードを入力します。	「*キーワード検索*」または「*キーワード選択*」を参照してください。
Command	ユーティリティコマンドを入力するコマンド行。	ユーティリティコマンドを入力します。「*ユーティリティコマンド*」を参照してください。

手順 54.3. カテゴリ検索

■ アスタリスク（*）を入力するか、オプションでアスタリスクで区切った接頭辞をカテゴリ検索フィールドに入力します。例：

```
N*
```

これで、次のようなメニューが表示されます。

```
        Search for Categories

 Mark Category
 ---- N*____________________________
   _  NATURAL ENVIRONMENT
   _  NATURAL OBJECTS
```

特定のカテゴリを選択条件として選択するには、任意の文字を［**Mark**］列に入力します。これで、この選択条件を満たすすべての API のリストが表示されます。

手順 54.4. キーワード検索

■ アスタリスク（*）を入力するか、アスタリスクで区切った接頭辞を入力します。例：

```
PF*
```

次のようなメニューが表示され、別のウィンドウに PF で始まるキーワードが表示されます。

```
        Search for Keywords

Mark Keyword
---- PF*_________________
  _  PF-KEY
  _  PF-KEY LINE
```

特定のキーワードを選択条件として選択するには、任意の文字を［**Mark**］列に入力します。これで、この選択条件を満たすすべての API のリストが表示されます。

手順 54.5. キーワード選択

■ キーワード全体を入力します。例：

```
PF-KEY
```

これで、キーワードが PF-KEY であるすべての API のリストが表示されます。

注意: 末尾以外のアスタリスクは文字どおりに解釈されます。例えば、*LANGUAGE を入力すると、キーワードが *LANGUAGE である API のリストが表示されます。

PF キー

次の PF キーを使用できます。

PF キー	Name	関数
PF1	**Help**	コンテキスト依存ヘルプを表示します。入力フィールドごとに特定のヘルプテキストがあります。コマンド行などのその他のコンテキストでは、一般的なヘルプテキストが表示されます。
PF2	**Reset**	すべての選択フィールドをクリアし、API リストを再調整します。
PF3	**Exit**	SYSEXT ユーティリティを終了するか、現在のメニューまたはウィンドウを閉じます。
PF4	**Asc/Desc**	API の昇順（**Asc**）と降順（**Desc**）を切り替えます。
PF5	**All/Curr**	すべての API を表示するメニュー（**All**）と、現在の API のみを表示するメニュー（**Curr**）を切り替えます。「インターフェイスバージョン」を参照してください。
PF6	**--**	リストの先頭までスクロールします。
PF7	**-**	1 ページ上方にスクロールします。
PF8	**+**	1 ページ下方にスクロールします。
PF9	**++**	リストの末尾までスクロールします。
PF10	**<**	最初のメニューに移動します。
PF11	**>**	2 番目のメニューに移動します。
PF12	**Canc**	SYSEXT ユーティリティを終了するか、現在のメニューまたはウィンドウを閉じます。

行コマンド

行コマンドを使用すると、オブジェクト操作を実行できます。行コマンドは、目的の API の横の［**Cmd**］列に入力できます。疑問符（?）を入力するか PF1 キーを押すと、有効な行コマンドがリストされます。

次の行コマンドを使用できます。

行コマンド	関数
K	指定されている API に関連するキーワードをリストします。
T	対応する API の説明のテキストオブジェクト USR*nnnn*T をリストします。 API と API に関連するキーワードおよびカテゴリの目的、機能、および呼び出し規則について説明します。
L	プログラム例 USR*nnnn*P をリストします。
E	プログラム例 USR*nnnn*P を編集します。
R	プログラム例 USR*nnnn*P を実行します。
X	プログラム例 USR*nnnn*P を実行します。
.	SYSEXT ユーティリティを終了します。

ユーティリティコマンド

このセクションでは、コマンド行で入力する次のユーティリティコマンドについて説明します。

- **EXIT**
  SYSEXT ユーティリティを終了します。
- **REFRESH**
  現在のライブラリ SYSEXT に含まれているオブジェクトのデータを使用して、API 情報を更新します。REFRESH コマンドは、API の説明またはキーワードが変更されたり、テキストオブジェクトが追加または削除されたりした場合にのみ必要となります。

  正常に完了すると、テキストモジュール EXT-XML1 および EXT-XML2 がライブラリ SYSEXT 内に生成されたことが確認されます。

  注意: ソースオブジェクトの EXT-XML1 と EXT-XML2 は変更しないでください。これらは、SYSEXT ユーティリティの設定に必須で、Software AG 内部でのみ使用されます。

インターフェイスバージョン

インターフェイスバージョンは、（ほぼ）機能が同じでも、拡張パラメータ仕様が異なる API のコレクションとみなすことができます。したがって、新しいバージョンと以前のバージョンの互換性を保つために、明確な開発サイクルに対応します。

API にインターフェイスバージョンがある場合は、対応するテキストオブジェクト USR*nnnn*T に表示されますインターフェイスバージョンは、それが属するバージョンに応じてリスト内で整列されます。右端の要素は現在のバージョンに属します。このステータスは、予約キーワード `+CURRENT-VERSION` で表されます。他のすべての要素は前のバージョンに属し、予約キーワード `+PREVIOUS-VERSION` でマークされています。インターフェイスバージョンを持たない API は、「ユニーク」な API と呼ばれています。

インターフェイスバージョンを持つ API は、メニュー上で強調表示されます。

現在の APIすべてのリスト（「**PF5**」を参照）は、すべてのユニークな API、およびインターフェイスバージョンを持つ API の現在のバージョン（キーワード `+CURRENT-VERSION` を使用）で構成されます。

予約キーワード

予約キーワードとは、API のメタ情報を指し、例えば API が追加された Natural バージョンなどが該当します。予約キーワードは常にプラス記号（+）で始まります。詳細については、次の表を参照してください。

予約キーワード	説明
+CURRENT-VERSION	インターフェイスバージョンを持つ API の現在のバージョン（「*インターフェイスバージョン*」を参照）。
+PREVIOUS-VERSION	インターフェイスバージョンを持つ API の以前のバージョン（「*インターフェイスバージョン*」を参照）。
+NEW-*PROD-version*	特定の製品に特定のバージョンで追加された API。例えば、`+NEW-NAT-6.3.11` は、バージョン `6.3.11` で製品 Natural に追加された API を指します。
+MOD-*PROD-version*	特定の製品に属し、特定のバージョンで変更された API。例えば、`+MOD-NAT-6.3.12` は、製品 Natural に属し、バージョン `6.3.12` で変更された API を指します。

Natural API の使用

システムライブラリ SYSEXT に含まれる Natural API を使用する場合は、次のいずれかの手順を実行します。

- この API を使用する Natural オブジェクトが含まれるユーザーライブラリの steplib ライブラリとして、システムファイル FNAT のシステムライブラリ SYSEXT を定義します。したがって、Natural バージョンをアップグレードする場合は、API 固有のアクションは必要ありません。
- 目的の API をシステムファイル FNAT のシステムライブラリ SYSTEM にコピーします。したがって、Natural バージョンをアップグレードする場合は、1 つのライブラリで API をチェックするだけで済みます。
- 目的の API をシステムファイル FUSER のシステムライブラリ SYSTEM にコピーします（非推奨）。
- 目的の API を、この API を使用する Natural オブジェクトが格納されているシステムファイル FUSER のユーザーライブラリ（またはその steplib の 1 つ）にコピーします（非推奨）。

API は、その API が提供されている Natural バージョンでのみ使用できます。API は FNAT システムファイルにのみ保存することを強くお勧めします。これにより、正しいバージョンが常に実行されます。

手順 54.6. インターフェイスを使用するには

1 呼び出し元のプログラムで `DEFINE DATA` ステートメントを使用して、その API のテキストオブジェクト `USRnnnnT` にリストされているパラメータを指定します。プログラム例 `USRnnnnP` では、`DEFINE DATA LOCAL` ステートメント内でパラメータを定義しています。または、呼び出し元のプログラム外の別の LDA（ローカルデータエリア）または PDA（パラメータデータエリア）で、そのデータエリアを参照する `DEFINE DATA LOCAL USING` ステートメントを使用してパラメータを指定できます。

2 次のステートメントを入力します。

```
CALLNAT 'USRnnnnN' parameters
```

詳細については、『ステートメント』ドキュメントの「*CALLNAT*」ステートメントを参照してください。

注意: 非標準の使用方法は、常にそれぞれのテキストオブジェクト `USRnnnnT` で記述します。

システムライブラリ SYSEXT に含まれている Natural コピーコードを使用する場合は、次の手順を実行します。

- 目的のコピーコードを、このコピーコードを使用する Natural オブジェクトが格納されているシステムファイル FUSER のユーザーライブラリにコピーします。

手順 54.7. コピーコードを使用するには

1. 呼び出し元のプログラムで `INCLUDE` ステートメントを使用して、その API のテキストオブジェクト `USRnnnnT` またはコピーコード自体にリストされているパラメータを指定します。プログラム例 `USRnnnnP` では、`DEFINE DATA LOCAL` ステートメント内でパラメータを定義しています。または、呼び出し元のプログラム外の別の LDA（ローカルデータエリア）または PDA（パラメータデータエリア）で、そのデータエリアを参照する `DEFINE DATA LOCAL USING` ステートメントを使用してパラメータを指定できます。
2. 一部のコピーコードでは、追加のデータ定義が必要です。これらは API のテキストオブジェクト `USRnnnnT` とコピーコード自体に記述します。
3. 次のステートメントを入力します。

```
INCLUDE USRnnnnX 'parameter'...
```

詳細については、『ステートメント』ドキュメントの「*INCLUDE*」ステートメントを参照してください。

注意: 非標準の使用方法は、常にそれぞれのテキストオブジェクト `USRnnnnT` で記述します。

VIII SYSEXV ユーティリティ

55 SYSEXV ユーティリティ

- 現行バージョンのプログラム例の実行 332
- 任意のバージョンのプログラム例の実行 332
- PF キー 333
- SYSEXV ユーティリティの終了 333

SYSEXVユーティリティを使用すると、現行および以前の一部バージョンのNaturalで使用可能な新機能の例にアクセスできます。すべてのプログラムはソースオブジェクトとして使用でき、実行時に詳細な機能の説明が表示されます。現行ではないNaturalバージョンのプログラム例も使用できます。

この章では、次のトピックについて説明します。

現行バージョンのプログラム例の実行

手順 55.1. 現行バージョンのプログラム例を実行するには

1 次のシステムコマンドを入力します。

```
SYSEXV
```

または:

SYSEXV ライブラリにログオンします。

```
LOGON SYSEXV
```

そして、次のように入力して続行します：

```
SYSEXV
```

これによって、メニューが表示され、Natural バージョンとカテゴリがリストされます。

2 メニューからバージョンを選択します。

これによって、コメント付きメニューが表示され、選択したバージョンに関連するプログラム例を選択できます。

3 プログラムを選択します。

これによって、プログラムが実行されます。

任意のバージョンのプログラム例の実行

手順 55.2. 任意のバージョンのプログラム例を実行するには

1 次のシステムコマンドを入力します。

```
SYSEXV
```

または:

SYSEXV ライブラリにログオンします。

```
LOGON SYSEXV
```

そして、次のように入力して続行します：

```
SYSEXV
```

メニューが表示され、Natural バージョンとカテゴリがリストされます。

2 メニューからカテゴリを選択します。

コメント付きメニューが表示され、選択したカテゴリに関連するすべてのプログラム例がリストされます。

3 プログラムを選択します。

プログラムが実行されます。

PF キー

次の PF キーを使用できます。

PF キー	Name	関数
PF1	Help	ヘルプテキストを表示します。
PF3	Exit	SYSEXV ユーティリティを終了するか、現在のメニューまたはウィンドウを閉じます。
PF5	すべて	現在のページのすべての項目を選択します。
PF7	Prev	1 ページ上方にスクロールします。
PF8	Next	1 ページ下方にスクロールします。
PF12	Canc	SYSEXV ユーティリティを終了します。

コマンド行に PF キーの名前を入力して、対応する機能を実行することもできます。

SYSEXV ユーティリティの終了

手順 55.3. SYSEXV ユーティリティを終了するには

■ SYSEXV ユーティリティメニューで、［**Exit**］を選択します。

または:

PF3 キーまたは PF12 キーを押します。

または:

任意の行にピリオド（.）を入力します。

または:

コマンド行で「EXIT」またはピリオド（.）を入力します。

IX SYSMAIN ユーティリティ - オブジェクトメンテナンス

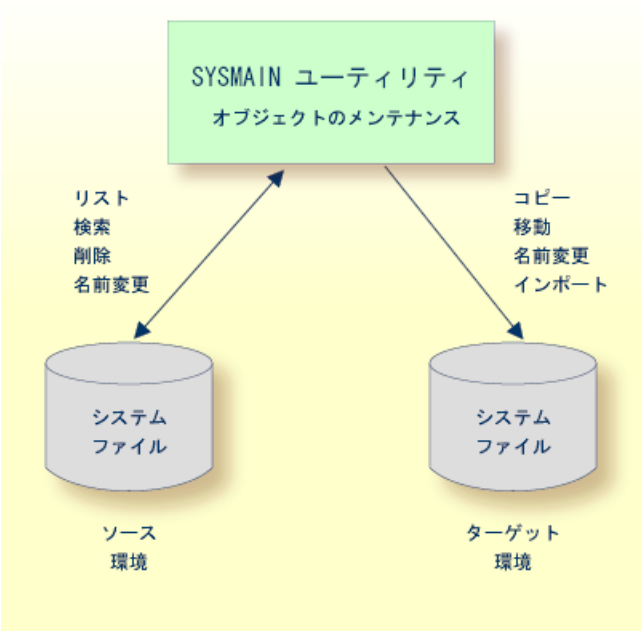
SYSMAIN ユーティリティは、コピー、移動、置換、削除などのオブジェクトメンテナンス機能を実行するために使用します。

全般的な情報	SYSMAIN の基本的な機能。
SYSMAIN の呼び出しと終了	SYSMAIN ユーティリティを呼び出して終了します。
機能の説明	SYSMAIN によって提供される機能を説明します。
オブジェクトのリストと検索	単一または複数ライブラリのオブジェクトをリストおよび検索します。
オブジェクトのコピー、移動、および名前変更	メニューモードで、ソースライブラリからターゲットライブラリへオブジェクトをコピーおよび移動します。ターゲット環境で、コピーされたオブジェクトに新しい名前を指定します。
オブジェクトの削除	メニューモードで、ソースライブラリからオブジェクトを削除します。
オブジェクトのインポート	メニューモードで、外部ソースから Natural ライブラリへオブジェクトをコピーします。
オブジェクト指定ウィンドウのフィールドの使用	メニューモードで、処理するオブジェクトの選択条件を指定します。
サブプログラムでの **SYSMAIN** の使用	オンラインまたはバッチモードのサブプログラムで SYSMAIN ユーティリティ機能を実行します。
XRef について	SYSMAIN を使用してクロスリファレンス（XRef）データを処理します。

56 全般的な情報

SYSMAIN ユーティリティを使用して、Natural 環境の Natural オブジェクトを管理します。

次の図は、SYSMAIN 機能を簡単な図で示したものです。



SYSMAIN ユーティリティでは、ソース環境からターゲット環境にオブジェクトをコピー、移動、またはインポートしたり、ソース環境で削除などのオブジェクト操作を実行したりできます。ソース環境またはターゲット環境とは、FNAT または FUSER システムファイルのライブラリです。

SYSMAIN ユーティリティの機能は、オンラインおよびバッチモードで使用できます。

SYSMAIN ユーティリティで管理できる Natural オブジェクトは、プログラミングオブジェクトおよびデータ定義モジュール（DDM）です。プログラミングオブジェクトは、プログラム、サ

ブプログラム、サブルーチン、マップ、テキスト、データエリア（ローカル、パラメータ、およびグローバル）、コピーコード、ヘルプルーチン、ダイアログ、クラス、機能、およびアダプタの各オブジェクトタイプで構成されています。

57 SYSMAIN の呼び出しと終了

- SYSMAIN の呼び出し 340
- SYSMAIN の終了 341

このセクションでは、メニューモードまたはサブプログラムを使用した SYSMAIN ユーティリティの呼び出し方法および終了方法について説明します。

SYSMAIN の呼び出し

以下の手順では、メニュー機能またはサブプログラムを使用して SYSMAIN ユーティリティを呼び出す方法について説明します。

手順 57.1. Natural メインメニューから SYSMAIN を呼び出すには

1 ［**Services**］を選択してから、［**SYSMAIN**］を選択するか、または「S」を入力します。

または:

［**Direct**］を選択し、［**Direct Command**］ウィンドウに以下のシステムコマンドを入力します。

```
SYSMAIN
```

次のような SYSMAIN メニューが表示されます。

```
2007-04-04                          NATURAL                  Library: SYSTEM
 19:56:22                  V 6.3.2 Software AG 2007         Mode   : STRUCTURED
 User: SAG                                                 Work Area : empty
+-----------------------------------------------------------------------------+
¦Library          Direct           Services         OS                Fin     ↩
 ¦
+-----------------------------------------------------------------------------+


                                  +-----------+
                                  ¦L List     ¦
                                  ¦F Find     ¦
                                  ¦C Copy     ¦
                                  ¦M Move     ¦
                                  ¦D Delete   ¦
                                  ¦R Rename   ¦
                                  ¦I Import   ¦
                                  ¦T Terminate¦
                                  +-----------+



  List all files in Library
```

システム変数 *LIBRARY-ID の現在の設定が SYSMAIN に渡され、オブジェクト処理のデフォルトのソースライブラリとして使用されます（上記の例では SYSTEM）。

2 機能を選択するか、または目的の機能に対応する 1 文字のコードを入力します。例えば、［**List**］の場合は「L」を入力します。

［**Import**］および［**Terminate**］以外は、次のような選択ウィンドウが表示されます。

```
+------- LIST --------+
¦X Programming Objects¦
¦  Views (DDM)        ¦
+--------------------- + ↵
```

3 DDM 以外のオブジェクトタイプを処理する場合は、必要に応じて UP ARROW キーを押して［**Programming Objects**］に印を付けます（デフォルト）。

または:

DDM のみを処理する場合は、DOWN ARROW キーを押して［**Views (DDM)**］に印を付けます。

手順 57.2. オンラインまたはバッチモードのサブプログラムで SYSMAIN を呼び出すには

■ CALLNAT ステートメントを含む Natural プログラムを使用します。このステートメントにより、SYSMAIN にコマンドを発行する MAINUSER サブプログラムが呼び出され、実行されます。「*サブプログラムでの SYSMAIN の使用*」も参照してください。

SYSMAIN の終了

手順 57.3. SYSMAIN を終了するには

■ ［SYSMAIN］メニューの［**Terminate**］を選択します。

または:

ESC キーを繰り返し押します。

または:

MAINUSER サブプログラムを使用している場合は、コマンド文字列にピリオド（.）を指定します。「*サブプログラムでの SYSMAIN の使用*」も参照してください。

58 機能の説明

次の表では、メニューモードまたは MAINUSER サブプログラムのコマンドとして使用できる SYSMAIN の機能について説明します。メニューモードで使用できる機能には、それぞれ同じ名前の対応コマンドがあります。例外：**Terminate**。

機能／コマンド	説明
List	指定したライブラリの Natural オブジェクトまたはオブジェクトの範囲をリストして、これらのオブジェクトのソースを表示するオプションを提供します。また、この機能を使用すると、現在の Natural システム環境に含まれるライブラリのリストも作成できます。
Find	単一ライブラリまたはライブラリの範囲にある Natural オブジェクトまたはオブジェクトの範囲を検索およびリストして、これらのオブジェクトのソースを表示するオプションを提供します。
Copy	Natural オブジェクトをソースライブラリからターゲットライブラリにコピーします。ソースライブラリのオブジェクトは何も変更されません。 コピーされるオブジェクトと同じ名前のオブジェクトがすでにターゲットライブラリに含まれている場合、指定したオブジェクトはコピーされません。ターゲットライブラリのオブジェクトを上書きする場合は、置換オプション（「*置換オプションの使用*」を参照）を使用します。
Move	Natural オブジェクトをソースライブラリからターゲットライブラリに転送します。オブジェクトはソースライブラリから削除され、ターゲットライブラリに追加されます。 コピーされるオブジェクトと同じ名前のオブジェクトがすでにターゲットライブラリに移動されている場合、指定したオブジェクトは移動されません。ターゲットライブラリのオブジェクトを上書きする場合は、置換オプション（「*置換オプションの使用*」を参照）を使用します。
Delete	Natural オブジェクトをソースライブラリから削除します。 メニューモードでは、確認オプションを使用して、削除を確定するか、または取り消すことができます。

機能／コマンド	説明
Rename	次のいずれかの方法を使用して、オブジェクトに新しい名前を付けます。 1. ソースライブラリでオブジェクトまたはオブジェクトの範囲の名前を変更します。 2. オブジェクトまたはオブジェクトの範囲をソースライブラリからターゲットライブラリにコピーして、ターゲットライブラリで名前を変更します。 名前が変更されるオブジェクトと同じ名前のオブジェクトがすでにライブラリに含まれている場合、もともと存在するオブジェクトの名前については変更されません。もともと存在するオブジェクトを上書きする場合は、置換オプション（「*置換オプションの使用*」も参照）を使用します。
Import	インポート機能を使用して、外部ソースからNaturalライブラリにオブジェクト（ファイル）をコピーします。または、オブジェクトハンドラ（関連ドキュメントを参照）のアンロード機能とロード機能を使用することもできます。 オブジェクトをインポートする前に以下の点を考慮する必要があります。 **FILEDIR.SAG** ： *FILEDIR.SAG* には、オブジェクト名、プログラミングモード（ストラクチャードまたはレポーティング）、オブジェクトの種類（ソースオブジェクトおよびカタログ化オブジェクトのいずれかまたはその両方）、およびユーザー ID など、Natural で必要な内部ライブラリ情報が含まれています。オブジェクトをインポートすると、ターゲットライブラリのファイルディレクトリ *FILEDIR.SAG* が自動的に更新され、新たにインポートしたオブジェクトの情報が含まれるようになります。 Natural以外の機能を使用してオブジェクトをNaturalライブラリにコピーした場合、Naturalではファイルディレクトリ *FILEDIR.SAG* が更新*されない*ことに注意してください。したがって、このライブラリに含まれるオブジェクトにアクセスすることはできません。 SYSMAIN でインポートするオブジェクトは、Natural で作成しておく必要があります。
Terminate	SYSMAIN ユーティリティを終了します。

メニュー機能を使用するか、または MAINUSER サブプログラムで SYSMAIN に発行されるコマンドを使用して、SYSMAIN 機能を実行する手順については、このドキュメントの関連するセクションを参照してください。プログラミングオブジェクトの処理とDDMの処理は似ているため、メニュー機能の手順ではプログラミングオブジェクトの選択についてのみ説明します。

59 オブジェクトのリストと検索

このセクションでは、メニューモードでリスト機能および検索機能を使用する手順について説明します。

手順 59.1. 単一または複数のオブジェクトを検索またはリストするには

1 SYSMAIN メニューで、［**List** ］を選択（または「L」を入力）するか、または［**Find** ］（または「F」を入力）を選択してから、［**Programming Objects**］を選択します。

次のようなオブジェクト指定ウィンドウが表示されます。

```
+--------------------- LIST ----------------------+
¦                                                 ¦
¦ OBJECT:  *                                      ¦
¦ LIBRARY: SYSTEM          CODE: X (S)ource       ¦
¦ DBID:     99   FNR:  51        X (C)ataloged    ¦
¦ USER ID:                                        ¦
¦ DATE:        -  -     :                         ¦
+-------------------------------------------------+
```

2 処理するオブジェクトを選択するための選択条件を入力します。このウィンドウに表示されるフィールドと有効な入力値については、「*オブジェクト指定ウィンドウのフィールドの使用*」を参照してください。

3 次の手順は、リスト機能にのみ適用されます。

- ［**LIBRARY**］フィールドに名前の範囲を入力した場合は、指定したシステムファイル内にある指定した範囲内のすべてのライブラリが選択用にリストされます。以下の例では、TEST* が入力され、名前が TEST から始まるすべてのライブラリがリストされています。

```
+-------- LIST --------+
¦  Library   dbid/fnr  ¦
¦ -------------------- ¦
¦  TEST       99/51    ¦
¦  TESTLIB1   99/51    ¦
¦  TESTLIB2   99/51    ¦
¦  TESTLIB3   99/51    ¦
¦ ___________________ ¦
¦ *** ENTER==>list *** ¦
¦ ***   ESC==>exit *** ¦
+----------------------+
```

- UP ARROW キーまたは DOWN ARROW キーを押してリストを上下にスクロールし、ENTER キーで必要なライブラリを選択します。

  または、
  何も行わずにウィンドウを閉じる場合は、ESC キーを押します。

ライブラリを選択するとオブジェクト指定画面が開き、選択したライブラリが［**LIBRARY**］フィールドに表示されます。

4 オブジェクト指定ウィンドウに選択条件を入力し、ENTER キーで入力を確定すると、以下の例のような［**OBJECT TYPE**］ウィンドウが表示されます。

```
+---OBJECT TYPE---+
¦X ==> select ALL ¦
¦   Program       ¦
¦   Subroutine    ¦
¦   Copycode      ¦
¦   Map           ¦
¦   Text          ¦
¦   Helproutine   ¦
¦   Subprogram    ¦
¦   Global Data   ¦
¦   Local Data    ¦
¦   Parameter Data¦
¦   Dialog        ¦
¦   Class         ¦
¦   Function      ¦
¦   Adapter       ¦
+-----------------+
```

5 このウィンドウにリストされているすべてのタイプのオブジェクトを処理する場合は、［**select ALL**］に「X」で印を付けます（デフォルト）。

または:

特定の1つまたは複数のタイプのオブジェクトのみを処理する場合は、必要なタイプの隣の入力フィールドに「X」を入力して、ENTER キーで確定します。

結果ウィンドウが表示されます。［**OBJECT**］フィールドに指定した単一のオブジェクト、または名前の範囲に含まれるすべてのオブジェクトが、以下の例のようにリストされます。

```
+---- 10 Object(s) in Lib: TESTLIB----+
¦ Object   Type         S/C  User ID  ¦
¦ -------- ------------ ---  -------- ¦
¦ EMPL-LDA Local        S    SAG      ¦
¦ MAP2     Map          S    SAG      ¦
¦ MAP3     Map          S/C  SAG      ¦
¦ PGM1     Program      S    SAG      ¦
¦ PGM2     Program      S    SAG      ¦
¦ PGM3     Program      S    SAG      ¦
¦ SUBPGM2  Subprogram   S/C  SAG      ¦
¦ SUBPGM3  Subprogram   S/C  SAG      ¦
¦ ____________________________________ ¦
¦ *** ENTER==>list *** ESC==>exit *** ¦
+-------------------------------------+
```

ウィンドウのタイトルに、検索されたオブジェクトの数（上記の例では10）および検索されたライブラリ（上記の例ではTESTLIBなど）が示されます。リストされているオブジェクトごとに、オブジェクト名、オブジェクトタイプ、利用可能なオブジェクトの種類（Sはソースオブジェクト、Cはカタログ化オブジェクト）、およびオブジェクトを保存またはカタログ化したユーザーの ID がウィンドウに表示されます。オブジェクトリストは、オブジェクト名のアルファベット順にソートされます。

以下は検索機能にのみ適用されます。

［**LIBRARY**］フィールドに名前の範囲を入力した場合、目的のオブジェクトが見つかったライブラリごとに結果ウィンドウが表示されます。ESC キーを押すと、アルファベット順の次のライブラリの結果ウィンドウが開きます。

6 オブジェクトのソースコードを表示する場合、そのオブジェクトのソースオブジェクトが存在する場合には、リストから必要なオブジェクトを選択して ENTER キーを押します。

［**List**］ウィンドウが開き、以下のプログラム PGM3 の例のように、選択したオブジェクトのソースコードが表示されます。

```
+------------------------------- List: PGM3 ---------------------------------+
¦ 0010 ********************************************************************** ¦
¦ 0020 * EXAMPLE:    'PGM3': AT BREAK STATEMENT                               ¦
¦ 0030 *                                                                      ¦
¦ 0040 * PURPOSE:    DEMONSTRATE NATURAL SYSTEM FUNCTIONS WITH AT BREAK       ¦
¦ 0050 *             CONDITION. INCLUDE USER-SUPPLIED TEXT.                   ¦
¦ 0060 *                                                                      ¦
¦ 0070 * HIGHLIGHTS: AT BREAK STATEMENT, NATURAL SYSTEM FUNCTIONS OLD, MIN,    ¦
¦ 0080 *             AVER, MAX, SUM, TOTAL, COUNT                             ¦
¦ 0090 ********************************************************************** ¦
¦ 0100 DEFINE DATA                                                            ¦
¦ 0110    LOCAL                                                               ¦
¦ 0120 1 EMPLOY-VIEW VIEW OF EMPLOYEES                                        ¦
¦ 0130   2 NAME                                                               ¦
¦ 0140   2 CITY                                                               ¦
¦ 0150   2 SALARY (1)                                                         ¦
+-------------------------------------------------------------------------------+
```

ESC キーを押して、このウィンドウを閉じます。

60 オブジェクトのコピー、移動、および名前変更

このセクションでは、メニューモードでコピー、移動、および名前変更の各機能を使用する手順について説明します。

手順 60.1. 単一または複数のオブジェクトをコピー、移動、または名前変更するには

1 SYSMAIN メニューで、［**Copy**］を選択（または「C」を入力）するか、［**Move**］を選択（または「M」を入力）するか、あるいは［**Rename**］を選択（または「R」を入力）してから、［**Programming Objects**］を選択します。

次のようなオブジェクト指定ウィンドウが表示されます。

```
+--------------------- COPY ----------------------+
¦                                                 ¦
¦                     - Source -                  ¦
¦ OBJECT:                                         ¦
¦ LIBRARY: SYSTEM          CODE: X (S)ource       ¦
¦ DBID:     99   FNR:  51        X (C)ataloged    ¦
¦ USER ID:                 XREF: n                ¦
¦ DATE:        -  -     :                         ¦
¦                                                 ¦
¦                     - Target -                  ¦
¦ OBJECT:                                         ¦
¦ LIBRARY:                 REPLACE:               ¦
¦ DBID:          FNR:                             ¦
+-------------------------------------------------+
```

2 ［**Source**］セクションに、ソース環境の指定に必要なすべてのオブジェクト選択条件を入力します。このウィンドウに表示されるフィールドと有効な入力値については、「*オブジェクト指定ウィンドウのフィールドの使用*」を参照してください。

3 ENTER キーで［**Source**］セクションの入力を確定します。

- ［**OBJECT**］フィールドおよび［**LIBRARY**］フィールドに単一の名前を入力して一致するオブジェクトが見つかった場合は、手順 **10** にスキップします。
- ［**LIBRARY**］フィールドに名前の範囲を入力した場合は、指定した範囲内のすべてのライブラリが選択用にリストされます（前に示したウィンドウの例を参照）。

4 必要なライブラリを選択します。

オブジェクト指定画面が開き、［**Source**］セクションの［**LIBRARY**］フィールドに、選択したライブラリが表示されます。

5 ENTER キーを押して続行します。

- ［**OBJECT**］フィールドに単一の名前を入力して一致するオブジェクトが見つかった場合は、手順 **10** にスキップします。
- ［**OBJECT**］フィールドに名前の範囲を入力した場合は、［**OBJECT TYPE**］ウィンドウ（前に示した図と説明を参照）が表示されます。

6 必要なオブジェクトタイプを 1 つまたは複数選択します。

7 コピーまたは移動機能を使用する場合は、以下の例のようなウィンドウが表示されます。

```
+-----------------------------------------+
¦X Select the specified Object(s) for copy ¦
¦  Copy  ALL  specified Object(s)          ¦
+-----------------------------------------+
```

一致するすべてのオブジェクトをリストからまとめて選択し、場合によっては選択したライブラリの名前をターゲットライブラリで変更する場合、DOWN ARROW キーを押して［**Copy ALL specified Object(s)**］に印を付けてから、手順 **10** にスキップします。

または、
一致するすべてのオブジェクトをリストから個々に選択し、場合によっては選択したライブラリの名前をターゲットライブラリで変更する場合、必要に応じて UP ARROW キーを押して［**Select the specified Object(s) for copy**］に印を付けます（デフォルト）。

- 単一のオブジェクトが見つかった場合は、手順 **10** にスキップします。
- 複数のオブジェクトが見つかった場合は、結果ウィンドウ（前に示した図と説明を参照）に似た選択ウィンドウが表示されます。

8 目的のオブジェクトの隣にある［**X**］列の入力フィールドに「X」を入力して、処理するオブジェクトを選択します。選択したオブジェクトにソースオブジェクトとカタログ化オブジェクトの両方が存在する場合、［**S/C**］列の S または C をそれぞれ空白文字で置き換えてから ENTER キーを押すと、ソースオブジェクトまたはカタログ化オブジェクトのみを処理できます。

9 目的のオブジェクトの選択が完了すると、以下のようなウィンドウが表示されます。

コピーまたは移動機能：

```
+----------------------------------------------------+
¦X Copy selected Objects using   same    Object name ¦
¦  Copy selected Objects using different Object name ¦
+----------------------------------------------------+
```

- ソースライブラリとターゲットライブラリの両方で同じオブジェクト名を使用する場合は、必要に応じて UP ARROW キーを押して最初のオプションに印を付けます（デフォルト）。
- または、
  ターゲットライブラリの各オブジェクトの名前を変更する場合は、DOWN ARROW キーを押して 2 番目のオプションに印を付けます。

名前変更機能：

```
+---------------------------------------------------+
¦X Rename every single Object with a new name     ¦
¦  Rename all selected Objects with one new name* ¦
+---------------------------------------------------+
```

- 各オブジェクトの名前を個別に変更する場合は、必要に応じて UP ARROW キーを押して最初のオプションに印を付けます（デフォルト）。
- または、
  オブジェクトの範囲の名前を新しい名前の範囲で変更する場合は、DOWN ARROW キーを押して 2 番目のオプションに印を付けます。

10 ［**XREF**］オプションが Y（Yes）に設定されている場合は、以下の例のようなウィンドウが表示されます。

```
+--------------------- XREF ----------------------+
¦                                                 ¦
¦          --- PREDICT (FDIC) Files ---           ¦
¦                                                 ¦
¦    - Source -              - Target -           ¦
¦ DBID:         0         DBID:         0         ¦
¦ FNR:          0         FNR:          0         ¦
¦ PASSWORD:               PASSWORD:               ¦
¦ CIPHER:                 CIPHER:                 ¦
+-------------------------------------------------+
```

11 必要に応じて、現在のデータベース ID（DBID）やファイル番号（FNR）、またはその両方を置き換えて、パスワードおよびサイファコードを入力します。詳細については、「*XRef について*」を参照してください。

12 オブジェクト指定画面が開きます。今度は以下の例のように表示されます。

```
+--------------------- COPY ---------------------+
¦ 3 Object(s) selected                           ¦
¦                     - Source -                 ¦
¦ OBJECT:  PRO*                                  ¦
¦ LIBRARY: TESTLIB         CODE: (S)ource        ¦
¦ DBID:    99    FNR: 51         (C)ataloged     ¦
¦ USER ID:                 XREF: ON              ¦
¦ DATE:        -  -     :  TYPE: all             ¦
¦                                                ¦
¦                      -Target -                 ¦
¦ OBJECT:  PRO*                                  ¦
¦ LIBRARY: TESTLIB2        REPLACE: n            ¦
¦ DBID:     99   FNR:  51                        ¦
+--------------------------------------------------+
```

- 複数のオブジェクトが見つかった場合、処理用に選択されたオブジェクトの数（上記の例では 3）がメッセージに示されます。
- オブジェクト名の範囲が［**Source**］セクションで指定され、名前変更オプション（手順 **9** を参照）が選択されていない場合、［**Target**］セクションの［**OBJECT**］フィールドは同じ名前の範囲を含む読み取り専用フィールドになります（上記の例ではPRO*）。

  単一のオブジェクトが処理用に選択されている場合は、このオブジェクトの名前が［**Target**］セクションの［**OBJECT**］フィールドに表示されます。このフィールドで名前を置き換えることができます。

  オブジェクト名の範囲が［**Source**］セクションで指定され、名前変更オプション（手順 **9** を参照）が選択されている場合、選択した各オブジェクトの名前が［**Target**］セクションの［**OBJECT**］フィールドに表示されます。このフィールドで、現在の各オブジェクトの名前を置き換えることができます。

  名前変更機能（手順**9** を参照）を使用すると、以下の例のような名前の範囲を指定することで複数のオブジェクトを置き換えることもできます。

```
+-------------------- RENAME --------------------+
¦ 3 Object(s) selected                           ¦
¦                - Old object name -             ¦
¦ OBJECT:  PRO*                                  ¦
¦ LIBRARY: TESTLIB2        CODE: (S)ource        ¦
¦ DBID:    99    FNR: 51         (C)ataloged     ¦
¦ USER ID:                 XREF: ON              ¦
¦ DATE:        -  -     :  TYPE: P               ¦
¦                                                ¦
¦                - New object name -             ¦
¦ OBJECT:  PGM*                                  ¦
¦ LIBRARY: TESTLIB2        REPLACE: n            ¦
¦ DBID:     99   FNR:  51                        ¦
+--------------------------------------------------+
```

上記の例では、名前が PRO で始まるすべてのオブジェクトが、名前が PGM で始まるオブジェクトと置き換えられます。

13 ENTER キーで［**Target**］セクションの入力を確定します。

指定した名前と同じ名前のオブジェクトがすでにターゲットライブラリに含まれていて、［**REPLACE**］オプションが N に設定されている場合、見つかったオブジェクトごとに以下のようなウィンドウが表示されます。

```
+-------------------------------------------------------------------------+
¦ Object(s) already exist, do you want to overwrite source ?   Y/N         ¦
+-------------------------------------------------------------------------+
```

- 「Y」（はい）を入力して、オブジェクトの置換を確定します。
- または、
  「N」（いいえ）を入力して、オブジェクトの置換を拒否します。

「*置換オプションの使用*」も参照してください。

14 コピー、移動、または名前変更の各機能が正常に終了すると、オブジェクト指定ウィンドウに適切な確認メッセージが表示されます。

61 オブジェクトの削除

このセクションでは、メニューモードで削除機能を使用する手順について説明します。

手順 61.1. 単一または複数のオブジェクトを削除するには

1 SYSMAIN メニューで、［**Delete**］を選択するか、または「D」を入力してから、［**Programming Objects**］を選択します。

次のようなオブジェクト指定ウィンドウが表示されます。

```
+-------------------- DELETE --------------------+
¦                                                ¦
¦                                                ¦
¦ OBJECT:                                        ¦
¦ LIBRARY: SYSTEM            CODE: X (S)ource    ¦
¦ DBID:     99   FNR:  51          X (C)ataloged ¦
¦ USER ID:                   XREF:    n          ¦
¦ DATE:        -  -     :    CONFIRM: y          ¦
+------------------------------------------------+
```

2 削除するオブジェクトの指定に必要な選択条件を入力します。このウィンドウに表示されるフィールドと有効な入力値については、「*オブジェクト指定ウィンドウのフィールドの使用*」を参照してください。

注意: XRef データはオブジェクトを削除すると常に削除されるため、［**XREF**］設定は無視できます。

3 ENTER キーでオブジェクト指定を確定します。

- ［**OBJECT**］フィールドおよび［**LIBRARY**］フィールドに単一の名前を入力して一致するオブジェクトが見つかった場合は、手順 **9** にスキップします。

- ［**LIBRARY**］フィールドに名前の範囲を入力した場合は、指定した範囲内のすべてのライブラリが選択用にリストされます（前に示したウィンドウの例を参照）。

4 必要なライブラリを選択します。

オブジェクト指定画面が開き、選択したライブラリが［**LIBRARY**］フィールドに表示されます。

5 ENTER キーを押して続行します。

- ［**OBJECT**］フィールドに単一の名前を入力して一致するオブジェクトが見つかった場合は、手順 **9** にスキップします。
- ［**OBJECT**］フィールドに名前の範囲を入力した場合は、［**OBJECT TYPE**］ウィンドウ（前に示した図と説明を参照）が表示されます。

6 必要なオブジェクトタイプを 1 つまたは複数選択します。

以下の例のようなウィンドウが表示されます。

```
+-------------------------------------------+
¦X Select the specified Object(s) for delete ¦
¦  Delete ALL specified Object(s)            ¦
+-------------------------------------------+
```

7 一致するすべてのオブジェクトをリストからまとめて選択する場合は、DOWN ARROW キーを押して［**Delete ALL specified Object(s)**］に印を付けてから、手順 **9** にスキップします。

または:

一致するすべてのオブジェクトをリストから個々に選択する場合は、必要に応じて UP ARROW キーを押して［**Select the specified Object(s) for delete**］に印を付けます（デフォルト）。

- 単一のオブジェクトが見つかった場合は、手順 **9** にスキップします。
- 複数のオブジェクトが見つかった場合は、結果ウィンドウ（前に示した図と説明を参照）に似た選択ウィンドウが表示されます。

8 削除するオブジェクトを選択します。

9 ［**CONFIRM**］オプションがデフォルトの Y（Yes）に設定されている場合は、以下の例のようなウィンドウが表示されます。

```
+-------------------------------------------------------------------------------+
¦ Are you sure you want to delete this Object ?   Y/N                           ¦
+-------------------------------------------------------------------------------+
```

- 「Y」（はい）を入力して、各オブジェクトの削除を確定します。
- または、
  「N」（いいえ）を入力して、各オブジェクトの削除を拒否します。

10 削除機能が正常に終了すると、オブジェクト指定ウィンドウに適切な確認メッセージが表示されます。

62 オブジェクトのインポート

このセクションでは、メニューモードで外部オブジェクト（ファイル）をインポートする手順について説明します。

オブジェクトをインポートする前に考慮する必要がある事項については、インポート機能の説明を参照してください。

手順 62.1. 単一または複数のオブジェクトをインポートするには

1 SYSMAIN メニューで、［**Import**］を選択するか、または「I」を入力します。

次のようなオブジェクト指定ウィンドウが表示されます。

```
+------------------------- IMPORT --------------------------+
¦                                                            ¦
¦                          - Source -                        ¦
¦                                                            ¦
¦ PATH:                                                      ¦
¦ OBJECT:              CODE: X (S)ource                      ¦
¦                            X (C)ataloged                   ¦
¦                                                            ¦
¦                          - Target -                        ¦
¦                                                            ¦
¦ LIBRARY:             USER ID:             REPLACE:        ¦
¦ DBID:                MODE:                                 ¦
¦ FNR:                                                       ¦
+------------------------------------------------------------+
```

2 ［**Source**］セクションに、ソース環境の指定に必要なすべてのオブジェクト選択条件を入力します。このウィンドウに表示されるフィールドと有効な入力値については、「*オブジェクト指定ウィンドウのフィールドの使用*」を参照してください。

3 ENTER キーで入力を確定します。

- ［**OBJECT**］フィールドに単一の名前を入力して一致するオブジェクトが見つかった場合は、手順 **5** にスキップします。
- ［**OBJECT**］フィールドに名前の範囲を入力した場合は、指定した範囲内のすべてのオブジェクトが以下の例のようにリストされます。

```
+------- Select for  IMPORT --------+
¦ X  Object   Type        S/C       ¦
¦ -  -------- ----------- --------- ¦
¦    CHECKSUB Subprogram  Source    ¦
¦    CHECKUSR Program     Source    ¦
¦    MAP-001  Map         Source    ¦
¦    STPCRES  Copycode    Source    ¦
¦    STPCSET  Copycode    Source    ¦
¦    STPLDATA Local Data  Source    ¦
¦    STPVERS  Subprogram  Source    ¦
¦    SYSDDM   DDM         Source    ¦
+-----------------------------------+
```

このウィンドウに含まれている列の説明については、前述の「結果ウィンドウ」を参照してください。

4 インポートするオブジェクトを選択します。

オブジェクト指定ウィンドウが開き、選択したオブジェクトの数が示されます。

5 ［**Target**］セクションに、必要なすべての指定を入力します。このウィンドウに表示されるフィールドと有効な入力値については、「*オブジェクト指定ウィンドウのフィールドの使用*」を参照してください。

インポートするオブジェクトと同じ名前のオブジェクトがすでにターゲットライブラリに含まれていて、［**REPLACE**］オプションが N に設定されている場合、確認ウィンドウが表示されます。「*オブジェクトのコピー、移動、および名前変更*」の手順 **13** で前述した指示に従います。

6 インポート機能が正常に終了すると、以下の例のような適切な確認メッセージが表示されます。

```
+------------------------- IMPORT --------------------------+
¦  3 Object(s) sele+----------------------+(s) imported     ¦
¦                  ¦ 3 Object(s) imported ¦                 ¦
¦                  +----------------------+                 ¦
¦ PATH:    /NAT/nathome/sag/                                ¦
¦ OBJECT:  STPVERS     CODE: Source                         ¦
¦ TYPE:    Subprogram                                       ¦
¦                                                           ¦
¦                          - Target -                       ¦
¦                                                           ¦
¦ LIBRARY: TESTLIB     USER ID: SAG         REPLACE: n      ¦
¦ DBID:     99         MODE:    X Structured                ¦
¦ FNR:      51                    Report                     ¦
+-----------------------------------------------------------+
```

63 オブジェクト指定ウィンドウのフィールドの使用

- 名前の範囲の指定 366
- 置換オプションの使用 367

このセクションでは、オブジェクト指定ウィンドウで指定可能なフィールドおよび入力オプションについて説明します。オブジェクト指定ウィンドウでは、SYSMAIN 機能で処理するオブジェクトの選択条件を指定できます。

フィールドが特定の機能のみに適用される場合は、適切な注釈が示されます。

フィールド	説明
OBJECT	処理するオブジェクトの名前または名前の範囲。 デフォルト設定はアスタリスク（*）であり、アスタリスクによってすべてのオブジェクトが処理用に選択されます。 有効な名前の範囲については、「*名前の範囲の指定*」を参照してください。
LIBRARY	ソースライブラリまたはターゲットライブラリの名前または名前の範囲。 ソースライブラリは、処理対象のオブジェクトを含みます。ターゲットライブラリは、オブジェクトのコピー先または移動先となる既存または新規のライブラリです。オブジェクトはここで名前変更またはインポートされることがあります。 有効な名前の範囲については、「*名前の範囲の指定*」を参照してください。
DBID	ソースライブラリまたはターゲットライブラリのデータベース ID。 ソースデータベースは、処理対象のオブジェクトが保存されているライブラリおよびシステムファイルを含みます。ターゲットデータベースは、オブジェクトのコピー先または移動先のライブラリおよびシステムファイルを含みます。オブジェクト名はここで変更されることがあります。 有効なデータベース ID は 1～65535 です。値を指定しない場合、または値が 0 の場合は、現在の FUSER または FNAT システムファイルが使用されます。
FNR	ソースシステムファイルまたはターゲットシステムファイル（FNAT または FUSER）のファイル番号。 ソースファイルは、処理対象のオブジェクトが保存されているライブラリを含みます。ターゲットファイルは、オブジェクトのコピー先または移動先のライブラリを含みます。オブジェクトはここで名前変更またはインポートされることがあります。 有効なファイル番号は 1～65535 です。値を指定しない場合、または値が 0 の場合は、現在の FUSER または FNAT システムファイルが使用されます。
USER ID	インポート機能には適用されません。 処理するオブジェクトを最後に保存またはカタログ化したユーザーの ID。
DATE	インポート機能には適用されません。 これらのフィールドに入力した日時以降に保存またはカタログ化されたすべてのオブジェクトを選択します。デフォルトでは、日時は入力されていません。 開始日は以下の形式で指定する必要があります。*YYYY-MM-DD*（*YYYY*=年、*MM*=月、*DD*=日）例：2007-01-31。 開始時刻は以下の形式で指定する必要があります。 *HH:II* （*HH*=時間、*II*=分）。例：09:15。
CODE	インポート機能には適用されません。

<table>
<tr><th>フィールド</th><th>説明</th></tr>
<tr><td></td><td>オブジェクトの種類を選択します。
<table>
<tr><td></td><td></td></tr>
<tr><td>(S)ource</td><td>ソース（保存）オブジェクトのみ。</td></tr>
<tr><td>(C)ataloged</td><td>カタログ化オブジェクトのみ。</td></tr>
<tr><td></td><td></td></tr>
</table>
デフォルトでは、ソースオブジェクトとカタログ化オブジェクトの両方が選択されます。

検索またはリスト機能のみ：
オブジェクトのソースコードは、ソースオブジェクトが存在する場合にのみ表示できます。［(C)ataloged］を選択した場合または結果ウィンドウにカタログ化オブジェクトのみが含まれている場合は、リストからオブジェクトを選択してソースコードを表示させることはできません。</td></tr>
<tr><td>XREF</td><td>検索、リスト、またはインポート機能には適用されません。

プログラミングオブジェクトの場合に Predict システムファイルに保存されているクロスリファレンス（XRef）データを処理するかどうかを指定します。これは DDM には適用されません。

使用可能な入力値は次のとおりです。
<table>
<tr><td></td><td></td></tr>
<tr><td>N</td><td>No。XRef データは処理されません。これはデフォルト設定です。</td></tr>
<tr><td>Y</td><td>Yes。すべての XRef データが処理されます。</td></tr>
<tr><td></td><td></td></tr>
</table>
「XRef の考慮事項」も参照してください。</td></tr>
<tr><td>TYPE</td><td>検索またはリスト機能には適用されません。

［OBJECT TYPE］ウィンドウで選択したオブジェクトタイプを示す読み取り専用フィールド。

フィールドには、すべてのオブジェクトタイプを示す all、Program などのオブジェクトタイプ（インポート機能のみ）、またはプログラムの場合は P などの 1 つまたは複数のオブジェクトタイプコードが含まれます。指定可能なコードについては、「TYPE の指定」を参照してください。</td></tr>
<tr><td>REPLACE</td><td>検索、リスト、または削除機能には適用されません。

置換オプションを有効にするかどうかを指定します。
<table>
<tr><td></td><td></td></tr>
<tr><td>Y</td><td>オブジェクトは自動的に置き換えられます。</td></tr>
<tr><td>N</td><td>オブジェクトは事前に確認した後でのみ置き換えられます。これはデフォルト設定です。</td></tr>
<tr><td></td><td></td></tr>
</table></td></tr>
</table>

フィールド	説明
	「*置換オプションの使用*」も参照してください。
CONFIRM	削除機能にのみ適用されます。 選択したオブジェクトを削除する前に、確認ウィンドウを表示するかどうかを示します。使用可能な入力値は次のとおりです。 `Y`：確認ウィンドウが表示されます。ここで「`Y`」を入力して削除を確定するか、または「`N`」を入力して削除を取り消すことができます。これはデフォルト設定です。 `N`：すべてのオブジェクトが即座に削除されます。事前の確認ウィンドウは表示されません。これはデフォルト設定です。
PATH	インポート機能にのみ適用されます。 インポート機能が実行されるディレクトリの完全な UNIX パス名。 パス名は、`$HOME` などの UNIX 環境変数で開始できます。ENTER キーを押すと、環境変数がフルパス名に置き換えられます。Natural セッションの開始時に割り当てられたデフォルトパスからオブジェクトをインポートする場合は、次を入力します。 `./`
MODE	インポート機能にのみ適用されます。 インポートするプログラミングオブジェクトに設定する Natural プログラミングモードを指定します。 **Structured**：ストラクチャードモードが使用されます。 **Report**：レポーティングモードが使用されます。 詳細については、『*プログラミングガイド*』の「*Natural プログラミングモード*」を参照してください。

名前の範囲の指定

すべての SYSMAIN 機能には、選択されるライブラリまたはオブジェクトについて名前または名前の範囲のいずれかを指定するオプションが用意されています。

次の表に、名前の範囲として有効なアスタリスク（*）表記を示します。*value* は 1 文字または複数文字の任意の組み合わせを示します。

入力	選択されるオブジェクトまたはライブラリ
*	すべてのオブジェクトまたはライブラリ。
*value**	名前が *value* で始まるすべてのオブジェクトまたはライブラリ。 例：AB* 選択対象：AB、AB1、ABC、ABEZ 選択対象外：AA1、ACB
*value*value**	1 つまたは 2 つのアスタリスク（*）と任意の順序で組み合わされた *value* に一致するすべてのオブジェクトまたはライブラリ。 例：A*C* 選択対象：ABCZ、AXXCBBBZ、ANCZ 選択対象外：ABDEZ、ACBBBZA

複数のオブジェクトの名前変更

複数のオブジェクトの名前を変更する場合は、ソース環境およびターゲット環境の両方に *value** を指定する必要があります。ソース環境に単一の名前を指定してターゲット環境に名前の範囲を指定することや、その逆はできません。

*value** を使用した場合、ソース環境のアスタリスク（*）の前の文字数で置換される文字数が決まります。例えば、ソース環境に ABC、ターゲット環境に WXYZ を指定した場合、ABC で始まるターゲット環境の各オブジェクトが WXYZ で始まるオブジェクト名に置き換えられます。各名前の残り（この例の場合、5 文字目以降）は維持されます。

置換オプションの使用

コピー、移動、名前変更、またはインポートされるオブジェクトと同じ名前のオブジェクトがすでにターゲットライブラリに含まれている場合、指定したオブジェクトは処理されず、次のオブジェクトの処理が続行されます。置換オプションを使用すると、このデフォルトの機能を無効にして、ターゲットライブラリのオブジェクトを上書きすることができます。オブジェクトが置き換えられた場合、そのオブジェクトは Natural バッファプールからも削除されます。既存のクロスリファレンスレコードもすべて削除されます。

手順 63.1. メニューモードで置換オプションを有効または無効にするには

- SYSMAIN オブジェクト指定ウィンドウの［**REPLACE**］フィールドに以下を入力します。

  置換オプションを有効にするには、「N」を入力します。

  各オブジェクトの置換を確認するプロンプトが表示されます。

  または:

置換オプションを無効にするには、「Y」を入力します。これは、デフォルトの設定です。

すべてのオブジェクトが、事前に確認メッセージを表示しないで置き換えられます。

手順 63.2. MAINUSER サブプログラムのコマンドを使用して置換オプションを有効にするには

■ 「*サブプログラムでの SYSMAIN の使用*」の説明に従って、コマンド文字列にキーワード REPLACE を指定します。

64 サブプログラムでの SYSMAIN の使用

- MAINUSER の呼び出しと実行 370
- コマンドの使用 370
- LIST および FIND コマンド構文 371
- COPY および MOVE コマンド構文 372
- DELETE コマンド構文 372
- RENAME コマンド構文 373
- IMPORT コマンド構文 373
- where-clause 374
- with-clause 374
- コマンドのキーワードと変数 374

MAINUSER サブプログラムは、一種のアプリケーションプログラミングインターフェイスです。ユーザーはSYSMAINユーティリティメニューを使わずに、ユーザー記述オブジェクト（サブルーチン、プログラム、またはサブプログラム）から SYSMAIN ユーティリティ機能を実行できるようになります。SYSMAIN 機能が完了した時点でユーティリティは終了し、制御は要求を発行したオブジェクトに戻されます。MAINUSERは、オンラインモードでもバッチモードでも使用することができます。呼び出し可能なルーチンの例として、SYSMAIN システムライブラリで提供される MAINCALL プログラムがあります。

このセクションでは、MAINUSER の使用手順に加えて、SYSMAIN ユーティリティ機能の実行用コマンドを指定するときに使用する構文について説明します。

MAINUSER の呼び出しと実行

手順 64.1. MAINUSER を呼び出して実行するには

■ 次の構文要素を指定して `CALLNAT` ステートメントを発行します。

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

ここでの各変数値は次のパラメータを表します。

パラメータ	Natural データフォーマット／データ長	説明
command	A250	SYSMAIN で実行するコマンド文字列の詳細については、「*コマンドの使用*」を参照してください。
error	N4	処理の最後に SYSMAIN によって発行される、処理の正常終了またはエラーを示すリターンコード。
message	A72	オンラインでのエラーに対応したメッセージ。
library	A8	ユーティリティ SYSMAIN を含むライブラリの名前。デフォルトでは、ライブラリ SYSMAIN がこれに当たります。（このパラメータは、互換性保持の目的でのみ提供されています）。

コマンドの使用

SYSMAIN 機能は、MAINUSER サブプログラムのパラメータとして発行するコマンドを使用することで実行できます。

*command*は、キーワードと変数値で構成されます。実行される SYSMAIN 機能ごとに、対応する構文図でキーワードと変数値を示し、「*コマンドのキーワードと変数*」で説明します。構文図の

記号は、システムコマンドで使用される構文記号に対応します。これらの記号の詳細については、『システムコマンド』ドキュメントの「システムコマンド構文」を参照してください。

コマンド構文の順序は必ずしも固定されていません。以下のルールが適用されます。

- SYSMAIN 機能、オブジェクトタイプ、およびオブジェクト名は、コマンド文字列の最初の 3 つのパラメータとして指定する必要があります。
- ピリオド（.）は、コマンドの終わりを示します。この文字がコマンド文字列内で検出されると、後続のデータはすべて無視されます。
- 構文図では、読みやすくするために FROM キーワードの代わりに FM または IN を使用していますが、常に FROM を FM または IN の同義語として使用でき、その逆も可能です。
- *where-clause* および *with-clause* の構文は、すべてのコマンドで同じです。

LIST および FIND コマンド構文

次のコマンド構文は、リスト機能および検索機能に適用されます。

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

例：LIST および FIND

```
LIST VIEW * IN TESTLIB
```

```
L SAVED TEST* IN TESTLIB TYPE PNS FNR 6
```

```
L SA TEST* IN TESTLIB FNR 6 DBID 2 TYPE PM FMDATE 2007-01-01
```

```
FIND PROG1 IN * DBID 1 FNR 6
```

```
F STOWED MAINMENU IN SYS* WHERE DBID 1 FNR 5
```

```
FIND ALL PROG2 IN PROD* FNR 27 DBID 1
```

COPY および MOVE コマンド構文

次のコマンド構文は、コピーおよび移動機能に適用されます。

```
{ COPY }  [ ALL       ]  name [ FM [LIBRARY] lib-name ]        [where-clause]
{ MOVE }  [ CATALOGED ]
          [ SAVED     ]
          [ STOWED    ]
          [ VIEW      ]        TO [LIBRARY] lib-name           [where-clause]
          [ RESOURCE  ]                                        [with-clause]
```

例：COPY および MOVE

```
COPY PROG1 FM TESTORD TO ORDERS DBID 1 FNR 6 REP

C PGM* FM TESTLIB TO PRODLIB WITH REP TYPE PNS

C VIEW PERS FM OLDLIB FNR 10 TO NEWLIB FNR 16 REPLACE

MOVE VIEW PERSONNEL FM OLDLIB FNR 20 TO NEWLIB FNR 24

M PROG1 TO NEWLIB

M STOWED * FM OLDLIB TO NEWLIB WHERE DBID 100 FNR 160 WITH XREF Y
```

DELETE コマンド構文

次のコマンド構文は、削除機能に適用されます。

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

例：DELETE

```
DELETE SA * IN LIBTEST TYPE GLA
```

```
D * IN TESTORD TYPE PM
```

```
D VIEW FINANCE IN TESTLIB DBID 12 FNR 27
```

RENAME コマンド構文

次のコマンド構文は、名前の変更機能に適用されます。

```
RENAME [ ALL | CATALOGED | SAVED | STOWED | VIEW | RESOURCE ]  name AS new-name [with-clause]
                                                                 FM [LIBRARY] lib-name [where-clause]
                                                                 TO [LIBRARY] lib-name [where-clause]
```

例：RENAME

```
RENAME PGM1 AS PROG1
```

```
R PGM1 AS PROG1 FM TESTLIB DBID 1 FNR 5 TO PRODLIB DBID 2 FNR 6
```

IMPORT コマンド構文

以下のセクションでは、インポート機能に適用されるコマンド構文を示します。

オブジェクトをインポートする前に考慮する必要がある事項については、インポート機能の説明を参照してください。

```
IMPORT [ ALL | CATALOGED | SAVED | STOWED | VIEW | RESOURCE ]  name FM [PATH] path-name
                                                                 TO [LIBRARY] lib-name [where-clause] [with-clause]
```

例：IMPORT

```
IMPORT ALL PGM* FM D:\NAT-PROGRAMS TO IMP-LIB
```

```
I RES res1.bmp FM D:\RESOURCES TO IMP-LIB
```

where-clause

```
[WHERE] [DBIDdbid] [FNRfnr]
                [DIC (dbid,fnr,password,cipher)]
                [SEC (dbid,fnr,password,cipher)]
```

セパレータ

DIC および SEC キーワードに続く各値の間に、または値がない場合に、セパレータとしてコンマを使用する必要があります。例えば、次のようになります。DIC (10,,secret,2a)。ID セッションパラメータ（『パラメータリファレンス』の「*ID - INPUT デリミタ文字*」も参照）がコンマに設定されている場合は、値を区切るセパレータとしてスラッシュ（/）を使用します。

with-clause

```
[WITH] [TYPEtype] [FMDATE date] [FMTIME time]
      [USER user-id]   [   XREF { Y | N } ]
      [REPLACE] [RCOP] [NOPROMPT] [HELP]
      [ STRUCT ]
      [ SM     ]
      [ REPORT ]
```

コマンドのキーワードと変数

このセクションでは、コマンドで使用するキーワードと、必要に応じて、対応する変数値について説明します。

キーワードはアルファベット順に示しています。斜体の文字はキーワードで指定する必要がある変数値です。変数値ごとに Natural データのフォーマットと長さが示されています。

キーワード	値	Natural データフォーマット／データ長	説明
ALL	*name*	A9	プログラミングオブジェクトにのみ適用されます。 処理されるオブジェクトの名前または名前の範囲（「*名前の範囲の指定*」を参照）。保存（ソース）オブジェクトおよびカタログ化オブジェクトのいずれかまたは両方が処理されます。
CATALOGED	*name*	A9	プログラミングオブジェクトにのみ適用されます。 処理されるカタログ化オブジェクトの名前または名前の範囲（「*名前の範囲の指定*」を参照）。
SAVED	*name*	A9	プログラミングオブジェクトにのみ適用されます。 処理される保存（ソース）オブジェクトの名前または名前の範囲（「*名前の範囲の指定*」を参照）。
STOWED	*name*	A9	プログラミングオブジェクトにのみ適用されます。 保存（ソース）オブジェクトおよびカタログ化オブジェクトが処理されるオブジェクトの名前または名前の範囲（「*名前の範囲の指定*」も参照）。保存（ソース）オブジェクトおよびカタログ化オブジェクトの両方として存在するオブジェクトのみが処理されます。 この例外はコピーコードとテキストで、両方ともカタログ化できません。ただし、このオプションを指定すると処理対象になります。
VIEW	*name*	A32	DDM にのみ適用されます。 処理される DDM の名前または名前の範囲（「*名前の範囲の指定*」を参照）。
RESOURCE	*name*	A255	共有リソースにのみ適用されます。 処理される共有リソースの名前または名前の範囲（「*名前の範囲の指定*」を参照）。
FROM または FM または IN	*lib-name* または *path-name*	A8 または A253	ソースライブラリまたはソースパスを指定します。 ソースライブラリまたはソースパスは、処理対象のオブジェクトを含みます。
TO	*lib-name*	A8	ターゲットライブラリを指定します。
AS	*new-name*	A8 または A32 または A255	RENAME コマンドで名前を変更するときに、オブジェクトに付ける新しい名前。フォーマット／長さは、プログラミングオブジェクトには A8、DDM には A32、共有リソースには A255 が適用されます。

キーワード	値	Natural データフォーマット／データ長	説明
LIBRARY	*lib-name*	A8	ソースライブラリまたはターゲットライブラリの名前（*lib-name*）を示すオプションキーワード。キーワードと各値を省略すると、SYSMAIN を呼び出す前にログオンしたライブラリが処理に使用されます。 ソースライブラリは、処理対象のオブジェクトを含みます。ターゲットライブラリは、オブジェクトのコピー先または移動先のライブラリです。オブジェクト名はここで変更されることがあります。 *lib-name* は、FROM/FM/IN または TO キーワードの直後に指定する必要があります。LIBRARY は、使用する場合は FROM/FM/IN または TO と *lib-name* の間に入力する必要があります。
PATH	*path-name*	A253	IMPORT コマンドにのみ適用されます。 ソースパスの名前（*path-name*）を示すオプションキーワード。有効なパス名については、「*オブジェクト指定ウィンドウのフィールドの使用*」の **PATH** を参照してください。 *path-name* は、FROM/FM/IN または TO キーワードの直後に指定する必要があります。PATH は、使用する場合は FROM/FM/IN または TO と *path-name* の間に入力する必要があります。
WHERE	*where-clause*	-	*where-clause* の開始を示すオプションキーワード。 *where-clause* は常に、FROM/FM/IN または TO キーワードおよび該当する場合はライブラリ名（*lib-name*）またはパス名（*path-name*）の後に入力する必要があります。節内のキーワードと値の順序は任意です。
DBID	*dbid*	N5	ソースシステムファイルまたはターゲットシステムファイルのデータベース ID（DBID）。 ソースシステムファイルは、処理対象のオブジェクトを含みます。ターゲットシステムファイルは、オブジェクトのコピー先または移動先のシステムファイルです。必要に応じてオブジェクト名が変更されます。 有効な DBID は 1～65535 です。 DBID または **FNR**（ファイル番号）を指定しない場合は、次の処理が適用されます。 現在のライブラリが含まれているシステムファイルの DBID と FNR が常に使用されます。 例：FUSER システムファイルに含まれているライブラリを指定した場合は、このシステムファイルの DBID と FNR が使用されます。

キーワード	値	Natural データフォーマット／データ長	説明
FNR	*fnr*	N5	ソースシステムファイルまたはターゲットシステムファイルのファイル番号（FNR）。 ソースシステムファイルは、処理対象のオブジェクトを含みます。ターゲットシステムファイルは、オブジェクトのコピー先または移動先のシステムファイルです。必要に応じてオブジェクト名が変更されます。 有効な FNR は 1～65535 です。 DBID（データベース ID）または FNR を指定しない場合は、次の処理が適用されます。 現在のライブラリが含まれているシステムファイルの DBID と FNR が常に使用されます。 例：FUSER システムファイルに含まれているライブラリを指定した場合は、このシステムファイルの DBID と FNR が使用されます。
DIC	*dbid* *fnr* *password* *cipher*	A80	データベース ID（*dbid*）、ファイル番号（*fnr*）、Adabas パスワード（*password*）、Adabas サイファコード（*cipher*）など、FDIC ソースシステムファイルとターゲットシステムファイルのいずれかまたは両方の環境を指定します。
SEC	*dbid* *fnr* *password* *cipher*	A80	データベース ID（*dbid*）、ファイル番号（*fnr*）、Adabas パスワード（*password*）、Adabas サイファコード（*cipher*）など、FSEC ソースシステムファイルとターゲットシステムファイルのいずれかまたは両方の環境を指定します。
WITH	*with-clause*	-	*with-clause* の開始を示すオプションキーワード。 *with-clause* のキーワードと値は任意の順序で指定でき、*with-clause* はコマンド文字列内の最初の 3 つの位置を除く任意の位置に配置できます。
TYPE	*type*	A20	以下の「*TYPE の指定*」でリストしている、処理対象オブジェクトのタイプ。
FMDATE	*date*	A10	期間の開始日。 指定した日付以降に保存またはカタログ化されたすべてのオブジェクトが処理されます。 日付は有効な Natural 日付フォーマットで指定する必要があります。デフォルトフォーマットは、2007-05-20 のように、国際標準フォーマットの *YYYY-MM-DD*（*YYYY* = 年、*MM* = 月、*DD* = 日）です。
FMTIME	*time*	A5	FMDATE を指定した場合にのみ適用されます。 開始時刻を指定します。

キーワード	値	Natural データフォーマット／データ長	説明
			指定した日時以降に保存またはカタログ化されたすべてのオブジェクトが処理されます。 時刻は次の形式で指定する必要があります：*HH*:*II*（*HH*=時間、*II*=分）。例：11:33。
USER	*user-id*	A8	ユーザー ID。 指定したユーザーによって保存またはカタログ化されたすべてのオブジェクトが処理されます。
XREF	N または Y	A1	Predict をインストールしている場合に、プログラミングオブジェクトにのみ適用されます。 Predict システムファイルに保存されたクロスリファレンス（XRef）データが処理されることを示します。 次の値のいずれかを指定できます。 N：DELETE コマンドを使用する場合を除いて、XRef データは処理されません。カタログ化オブジェクトを削除すると、このオブジェクトの既存の XRef データはすべて SYSMAIN により必ず削除されます。 Y：すべての XRef データが処理されます。 「*XRef の考慮事項*」も参照してください。
REPLACE	-	-	*with-clause* で使用されている置換オプションをアクティブ化します。 オブジェクトは自動的に置き換えられます。 「*置換オプションの使用*」も参照してください。
RCOP	-	-	名前を変更するオブジェクトのコピーを作成することを指定します。
NOPROMPT	-	-	バッチモードでは適用されません。 SYSMAIN プロンプトを無効化（NOPROMPT）します。 NOPROMPT の場合は、確認画面は表示されません。

キーワード	値	Natural データフォーマット／データ長	説明
			例えば、何かを削除する前に、SYSMAIN はユーザーに確認を求めます。
HELP	-	-	このキーワードは、互換性保持の目的でのみ提供されています。
STRUCT または SM			IMPORT コマンドにのみ適用されます。 ストラクチャードモード（『*プログラミングガイド*』の「*Natural プログラミングモード*」を参照）を示します。
REPORT			IMPORT コマンドにのみ適用されます。 レポーティングモード（『*プログラミングガイド*』の「*Natural プログラミングモード*」を参照）を示します。
.	-	-	ピリオド（.）は、コマンドの終わりを示します。この文字がコマンド文字列内で検出されると、後続のデータはすべて無視されます。

TYPE の指定

次の表に、TYPE キーワードで使用できるプログラミングオブジェクトの有効なオブジェクトタイプコードをすべて示します。

Code	オブジェクトタイプ
P	プログラム
N	Subprogram
S	サブルーチン
M	マップ
H	Helproutine
3	ダイアログ
5	プロセッサ
A	パラメータデータエリア
G	グローバルデータエリア
L	ローカルデータエリア
C	コピーコード
T	テキスト
4	クラス
7	関数
V	ビュー（DDM）

Code	オブジェクトタイプ
8	アダプタ
*	すべてのプログラミングオブジェクトタイプ

65 XRef について

- XREF を N に設定 382
- XREF を Y に設定 382
- FDIC ファイルセキュリティ 383
- XRef 処理のエラー 383

カタログ化プログラミングオブジェクト（DDM は適用外）の Predict システムファイルに格納されているすべてのクロスリファレンス（XRef）データを、SYSMAIN で処理できます。

Predict システムファイルは、パラメータファイル内または Natural セッション開始時のプロファイルパラメータ FDIC（『パラメータリファレンス』ドキュメントの「*FDIC - Predict システムファイル*」を参照）に割り当てられている値で決定されます。

現在実行中の SYSMAIN 機能の現在の FDIC 設定は、［**XREF**］ウィンドウの値（データベース ID およびファイル番号の 0 は現在の FDIC を示す）を置き換えるか、またはコマンドの *where-clause* に DIC キーワードを指定することによって上書きできます。

オブジェクト指定ウィンドウの［**XREF**］フィールドまたはコマンドの XREF キーワードによって、SYSMAIN で XRef データを処理するかどうかを指示します。

Predict がインストールされていない場合は、［**XREF**］フィールドに「N」を入力（または XREF=N を指定）して、Predict ファイルの XRef データが処理されないようにします。これはデフォルト設定です。

XREF を N に設定

［**XREF**］フィールドに N が含まれている場合（または XREF=N の場合）、XRef データは処理されません。

ただし、選択した設定にかかわらず、プログラミングオブジェクトを削除した場合は常に XRef データも削除されます。

XREF を Y に設定

［**XREF**］フィールドに Y が含まれている場合（または XREF=Y の場合）、XRef データは処理されます。

XRef データが処理される場合は、SYSMAIN 処理中に次の処理が適用されます。

- SYSMAIN により、指定したプログラミングオブジェクトの Predict ソースシステムファイルに XRef データが存在するかどうかが確認されます。
- プログラミングオブジェクトをターゲット環境から削除すると、XRef データが Predict ターゲットシステムファイルから削除されます。
- プログラミングオブジェクトを新しい環境にコピーすると、プログラミングオブジェクトの XRef データが Predict ソースシステムファイルから Predict ターゲットシステムファイルにコピーされます。それに応じてライブラリ名を名前の変更機能を使用して変更すると、オブジェクト名も変更されます。

- 移動機能を実行すると、プログラミングオブジェクトの XRef データが Predict ソースシステムファイルから削除されます。

FDIC ファイルセキュリティ

FDIC システムファイルにファイルセキュリティが設定されている場合は、SYSMAIN 機能を実行する前に、必要なソースシステムファイルとターゲットシステムファイルのいずれか、または両方のパスワードおよびサイファコードを指定する必要があります。指定しなかった場合、適切なエラーメッセージが表示されます。SYSMAIN ユーティリティの開始時に割り当てられるデフォルトシステムファイルに対してセキュリティ情報を提供する必要はありません。

XRef 処理のエラー

XRef データの SYSMAIN 処理中に次の矛盾が発生すると、オブジェクトまたは機能のすべての処理が終了し、エラーメッセージが表示されます。

- Natural Security の XREF オプションの値は `Y` に設定されていますが、［**XREF**］フィールドには `N` が含まれているか、または `XREF=N` が指定されています。
- ［**XREF**］フィールドに `Y` が含まれているか、または `XREF=Y)` が指定されていますが、使用されている FDIC ファイルは有効な Predict ファイルではありません。

X SYSNCP ユーティリティ

66 SYSNCP ユーティリティ

- UNIX の前提条件 388
- SYSNCP ユーティリティについて 388
- SYSNCP の呼び出し 394
- プロセッサの選択 396
- ヘッダーレコード 397
- キーワードのメンテナンス 406
- 機能のメンテナンス 412
- ランタイムアクション 417
- プロセッサのカタログ化 422
- 管理者サービス 423
- セッションプロファイル 430

SYSNCPユーティリティは、Naturalアプリケーションに対してコマンド方式のナビゲーションシステムを定義するために使用します。

Naturalコマンドプロセッサ（NCP）は、メンテナンスとランタイムの2つのコンポーネントで構成されています。SYSNCP ユーティリティはメンテナンス部分に相当し、アプリケーション内のナビゲーションを定義および制御するためのすべて機能を備えています。PROCESS COMMAND ステートメント（『ステートメント』ドキュメントを参照）は、Natural プログラムを呼び出すために使用されるランタイム部分です。

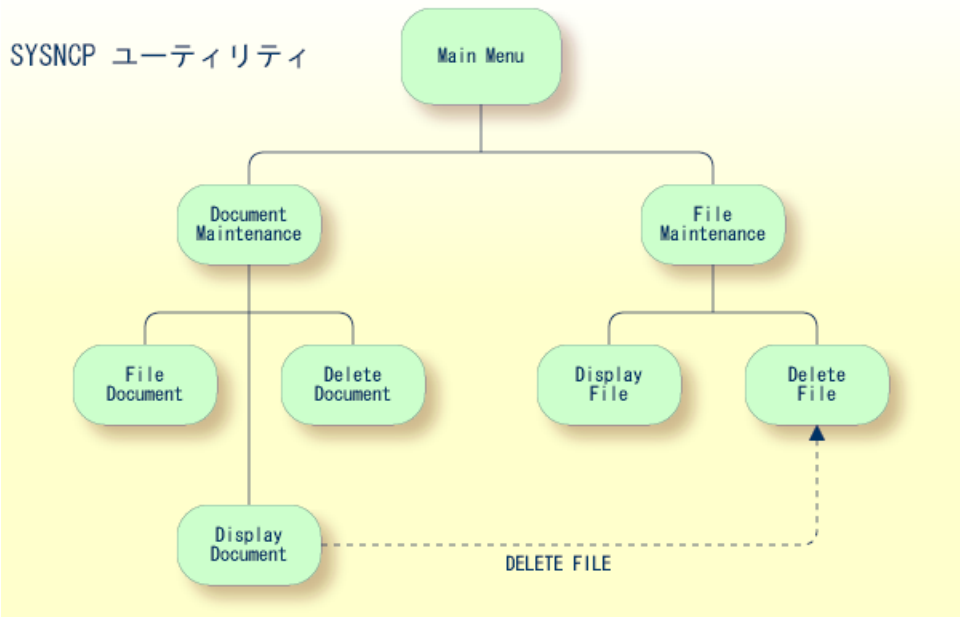
UNIX の前提条件

このセクションでは、UNIX環境にコマンドプロセッサをインストールするために必要な前提条件について説明します。

- 論理ファイル（LFILE）190（NCP Command Proc）。
- FDT "SYSTEM-NCP" がインストール時にロードされている必要があります。『*インストール*』ドキュメントの「*SYSPCI ユーティリティ - 製品の設定および初期化*」を参照してください。

SYSNCP ユーティリティについて

ユーザーがダイレクトコマンドを使用してあるアクティビティから別のアクティビティに移動できるアプリケーションは、ユーザーがメニュー階層を経由して目的のアクティビティに移動するアプリケーションよりも、有用性の点ではるかに上回っています。



上の図は、ダイレクトコマンドを使用することの利点を示しています。メニュー階層がナビゲーションの基盤となっているアプリケーションでは、Display Document（ドキュメント表示）機能から Delete File（ファイル削除）機能へ移動するユーザーは、ドキュメント関連のブランチを経由して Main Menu（メインメニュー）に戻ってから、ファイル関連のブランチに入る必要があります。これは、Display Document（ドキュメント表示）機能から Delete File（ファイル削除）機能に直接アクセスする場合と比べて、明らかに非効率的です。

以下に参考情報を示します。

- オブジェクト指向のデータ処理
- コマンドプロセッサの機能
- コマンドプロセッサの構成
- コマンドとは
- コマンドプロセッサの作成

オブジェクト指向のデータ処理

Natural コマンドプロセッサを使用して、アプリケーション内のナビゲーションを定義および制御します。例えば、コマンドプロセッサを使用して、Display Document（ドキュメント表示）機能に直接アクセスできる DISPLAY DOCUMENT コマンドを定義します。このコマンドを使用できる画面のコマンド行にこのコマンド文字列を入力すると、Natural コマンドプロセッサによって入力が処理され、コマンドに割り当てられているアクションが実行されます。

メニュー方式のアプリケーションとは対照的に、Natural コマンドプロセッサで実装されるコマンド方式のアプリケーションは、オブジェクト指向のデータ処理に向かって大きな一歩を踏み出しています。このアプローチには次の利点があります。

- アプリケーションの設計は、ある結果に達するまでの過程に依存する必要がなく、求める結果自体にのみ依存します。そのため、アプリケーション設計は構成要素内の処理フローに影響されなくなります。
- アプリケーションの処理ユニットが互いに独立するようになるため、簡単、迅速、しかもより効率的に、アプリケーションをメンテナンスすることができます。
- アプリケーションは、独立した処理ユニットを追加することで簡単に拡張できます。このため、エンドユーザーにとって利用しやすいだけでなく、プログラマにとっても作成しやすいアプリケーションになります。

Natural コマンドプロセッサには、さらに次の利点があります。

- プログラムコーディングの縮小
  今までは、コマンド処理を操作するために、長くて同じような構造化ステートメントブロックを繰り返す必要がありましたが、コマンドプロセッサを呼び出す PROCESS COMMAND ステートメントを指定するだけで済むようになりました。実際のコマンド操作をソースコードに指定する必要がなくなりました。これにより、必要なコーディングの量が大幅に削減されます。

- コマンド操作の効率化
  標準的な方法で一元集中的にコマンド操作を定義するため、アプリケーションのコマンド処理部分の作成およびメンテナンスに伴う作業をより迅速かつ効率的に行うことができます。
- パフォーマンスの改善
  Naturalコマンドプロセッサは、特にパフォーマンス面を考慮して設計されています。そのため、できる限り迅速にコマンドが処理され、Naturalアプリケーションのパフォーマンス改善に役立ちます。

コマンドプロセッサの機能

Naturalコマンドプロセッサには、効率的で使いやすいコマンド操作のための機能が数多く備えられています。

- コマンドの柔軟な操作
  頻繁に使用するコマンドに対して、別名（キーワードのシノニム）および省略形を定義できます。
- 省略されたキーワードの一意性の自動チェック
  コマンドプロセッサにより、SYSNCP に指定したキーワードごとに他のすべてのキーワードとの比較が自動的に行われ、キーワードを一意に識別するために必要な各キーワードの最小文字数が決定されます。つまり、アプリケーションでコマンドを入力する場合、ユーザーは各キーワードを、コマンドプロセッサが他のキーワードと区別するために必要とする最小限の長さに短縮できます。
- コマンドのローカルおよびグローバルな有効性
  特定のコマンドに対応して実行するアクションを、すべての条件で同じにするか、または状況に依存するかを SYSNCP で指定できます。例えば、前に発行されたプログラムに応じてアクションを決定することができます。さらに、ある条件下では有効で、別の条件下では無効になるようにコマンドを定義できます。
- 不正コマンドに対するエラー処理
  ユーザー独自のエラー処理ルーチンをコマンドに結び付けることや、エラー入力をNaturalで処理することができます。
- 機能のセキュリティ
  Natural Security を使用すると、ライブラリ固有およびユーザー固有の使用条件をSYSNCPで生成したテーブルに定義できます。このようにして、Naturalアプリケーションの特定の機能またはキーワードを、特定のユーザーに対して許可または禁止することができます。これを、機能のセキュリティと呼びます。『*Natural Security*』ドキュメントの「*Functional Security*」も参照してください。
- ヘルプテキスト
  SYSNCP では、キーワードまたはコマンドにヘルプテキストを結び付けることができます。その後で PROCESS COMMAND ACTION TEXT ステートメントを指定することにより、コマンド固有のヘルプテキストをプログラムに返すことができます。

- コマンド処理のオンラインテスト
  コマンドの実行結果が意図したものにならなかった場合は、PROCESS COMMAND ステートメント（『ステートメント』ドキュメントを参照）とライブラリ SYSNCP に提供されているソース形式のサンプルテストプログラム EXAM* を使用して、コマンドが正しく処理されなかった理由を調べることができます。EXAM-* プログラム名の末尾が、関連するアクションウィンドウの上の境界線に、省略形として表示されます。例えば、EXAM-C の場合は **C** と表示されます。

手順 66.1. コマンドプロセッサをランタイムでテストするには

1. ダイレクトコマンド「EXAM」を入力して、すべてのテストプログラムをリストします。［**Demonstrate PROCESS COMMAND Statement**］ウィンドウが表示されます。
2. ファンクションコード「**O**」を入力して、プロセッサを開きます。
3. プロセッサの名前を入力します。
4. リストされたファンクションコードのいずれかを選択して、コマンドアクションを適用します。例えば、CHECK の場合は C を選択します。
5. ファンクションコード「**Q**」を入力して、プロセッサを閉じます。

コマンドプロセッサの構成

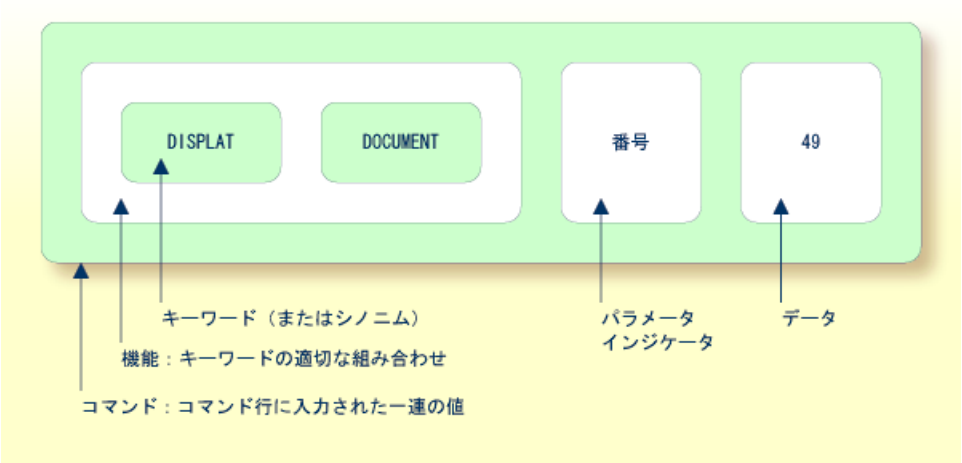
Natural コマンドプロセッサは、開発とランタイムの 2 つの部分で構成されます。

- 開発部分は SYSNCP ユーティリティであり、このセクションで説明しています。SYSNCP ユーティリティを使用して、コマンド（下記参照）およびコマンドの実行に対応して実行されるアクションを定義します。SYSNCP は、管理者が設定した定義により、コマンド入力時に実行する処理を判断するデシジョンテーブルを生成します。これらのテーブルは、プロセッサタイプの Natural メンバに含まれます。
- ランタイム部分は PROCESS COMMAND ステートメントであり、『ステートメント』ドキュメントで説明しています。PROCESS COMMAND ステートメントを使用して、Natural プログラム内でコマンドプロセッサを呼び出します。PROCESS COMMAND ステートメントでは、ユーザーによるコマンド入力を処理するためにこの時点で使用するプロセッサの名前を指定します。

コマンドとは

コマンドとは、コマンド行に入力される一連の値であり、アプリケーションによって認識および処理されます。コマンドは、最大 3 つの要素を含むことができます。

- 機能：
  1 つ以上の有効なキーワード。MENU または DISPLAY DOCUMENT など。
- パラメータインジケータ：
  指定は任意です。コマンドデータを導くキーワード。
- コマンドデータ：
  機能に送られる情報。コマンドデータは、英数字または数字で指定できます。例えば、表示されるファイルの名前または番号を指定します。



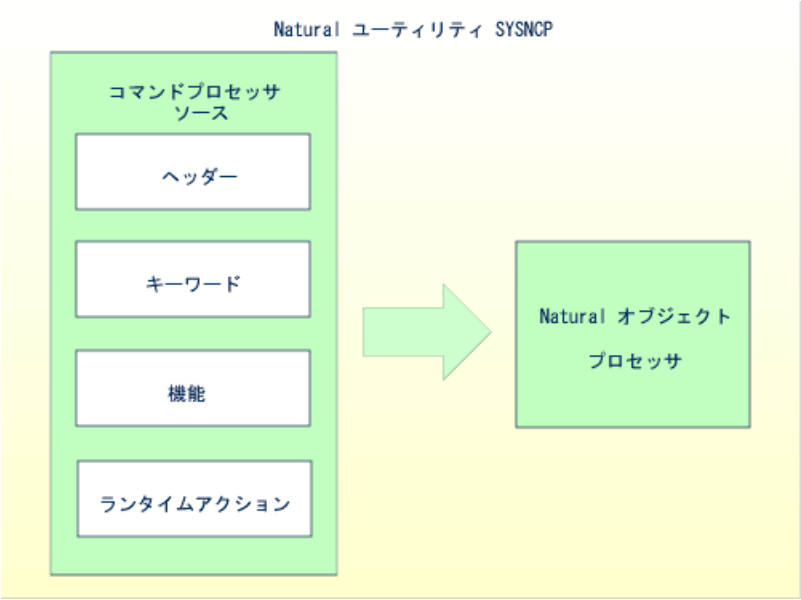
コマンドは、常にアプリケーション内のある状態から実行されます。この状態に達した位置をロケーションと呼びます。コマンドによって、ユーザーはあるロケーションから別のロケーションに導かれます。そのため、各コマンドはベクトルとみなすことができます。

特定のコマンドを発行できるロケーションを、システム全体またはユーザーベースで制限することができます。例えば、システム全体を基準とした場合、コマンド内に指定する機能をローカルまたはグローバルにすることができます。グローバル機能はあらゆるロケーションから発行できるのに対し、ローカル機能は指定されたロケーションからのみ発行できます。ただし、Natural Security がアクティブになっている環境では、キーワードおよび機能に制限を設けることができます。

コマンドプロセッサの作成

SYSNCP ユーティリティを使用して、コマンドプロセッサを作成およびメンテナンスします。コマンドプロセッサには、ユーザーが有効なコマンドを入力したときの処理を決定するデシジョンテーブルが含まれています。

コマンドプロセッサの作成は累積的な作業であり、プロセッサの全般的な設定を構築するヘッダー定義から、キーワード定義、機能定義、および機能に対するアクションの関連付けまでのさまざまな手順を伴います。SYSNCP には、キーワード、機能、およびアクションを指定するための特別なエディタが用意されています。



コマンドプロセッサ開発の最終的な成果物は複合的なコマンドプロセッサソースであり、このソースをカタログすると、プロセッサタイプの Natural オブジェクトが生成されます。このオブジェクトを Natural の PROCESS COMMAND ステートメントで参照するたびに、Natural コマンドプロセッサのランタイムシステムが起動されます。

コマンドプロセッサの作成に必要な手順の概要は、以下のとおりです。

手順 66.2. コマンドプロセッサを作成するには

1 セッションプロファイルを確認および修正します。
SYSNCP 自体が使用するセッションプロファイルには、SYSNCP による特定のアクションの実行方法および情報の表示方法を制御するさまざまなパラメータが含まれています。必要な修正を行った後、そのプロファイルを指定のユーザー ID で保存することができます。
「*セッションプロファイル*」を参照してください。

2 コマンドプロセッサを初期化します。
コマンドプロセッサの名前およびコマンドプロセッサを保存するライブラリを指定します。

3 グローバル設定（ヘッダー）を定義します。

コマンドプロセッサに対するさまざまなグローバル設定を定義します。例えば、キーワード編集時の記述テキスト、キーワードの最小文字数と最大文字数、ランタイム時に処理するキーワードの順番、ランタイムエラー処理、およびランタイム時の機能呼び出しに PF キーを使用できるかどうかを設定できます。「*ヘッダーレコード*」を参照してください。

4 キーワードを定義します。
コマンドプロセッサで処理するキーワードはそれぞれ、コマンドの 1～3 番目のどのエントリとして入力するかに関する指定とともに定義します。キーワードのシノニムは、パラメータインジケータとしても定義できます。キーワードごとにユーザーテキストを定義できます。このテキストは、その後、PROCESS COMMAND ACTION TEXT ステートメントを使用して、ランタイム時に読み込むことができます。「*キーワードのメンテナンス*」を参照してください。

5 機能を定義します。
機能は、キーワード結合を有効にすることで定義します。機能は、ローカル（アプリケーション内の特定のロケーションからのみ呼び出し可能）またはグローバル（アプリケーション内のどこからでも呼び出し可能）あるいはその両方として定義できます。「*機能のメンテナンス*」を参照してください。

6 ランタイムアクションを定義します。
ランタイム時にコマンドが発行されたときに、コマンドプロセッサによって実行されるアクションを定義します。例えば、Natural プログラムを取り出す、コマンドを Natural スタックの先頭に置く、データを Natural スタックの先頭に置く、およびコマンド行の内容を変更するアクションを定義します。「*ランタイムアクション*」を参照してください。

7 コマンドプロセッサをカタログします。
作成したソースを Natural オブジェクト（プロセッサタイプ）として、指定した Natural ライブラリにカタログします。これで、PROCESS COMMAND ステートメントを使用して、Natural プログラムからこのコマンドプロセッサを呼び出すことができます。「*プロセッサのカタログ化*」を参照してください。

SYSNCP の呼び出し

手順 66.3. SYSNCP ユーティリティを呼び出すには

■ システムコマンド SYSNCP を入力します。

［Processor Source Maintenance］メニューが表示されます。

```
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
```

このメニューから、コマンドプロセッサの作成およびメンテナンスに必要なすべての機能を呼び出すことができます。機能を呼び出すには、［Code］フィールドにコード文字を入力します。

注意: SYSNCP ユーティリティを呼び出すか、または SYSNCP を再起動すると、ダイナミックなカスタマイズを目的としてユーザー出口NCP-USR1が呼び出されます。Natural システムライブラリ SYSNCP で提供されているプログラム NCP-USR1 を参照してください。

Help

SYSNCPの各入力フィールドおよび一部の出力フィールドのヘルプを表示するには、そのフィールドにカーソルを置いて PF1 キーを押します。

プロセッサの選択

プロセッサ選択機能では、存在するすべてのコマンドプロセッサソースを関連情報とともにリストします。Natural Security がインストールされている場合、ログオンを許可されているライブラリに対してカタログ可能なソースのみがリストされます。これらの制限は、管理者ステータスを持つユーザーには適用されません。

手順 66.4. プロセッサ選択機能を呼び出すには

1 ［Processor Source Maintenance］メニューで、ファンクションコード「**S**」を入力します。

2 ENTER キーを押します。

プロセッサごとに次の情報が表示されます。

Name	コマンドプロセッサの名前。
ライブラリ	プロセッサを作成する Natural ライブラリの名前。プロセッサをカタログすると、このライブラリに保存されます。
User ID	プロセッサを作成したユーザーの ID。
日付	プロセッサが作成された日付。
ステータス	プロセッサ開発の段階。有効なステータスの値については、「*ヘッダーレコード*」の「***Current Status***」を参照してください。
Cat	プロセッサがカタログされているかどうかを示します。

注意: Natural システムライブラリ SYSNCP で提供されているユーザー出口 NCP-SELX を使用すると、表示内容を特定のプロセッサに制限することができます。

3 ［**Ac**］フィールドに任意の文字を入力して、プロセッサを選択します。

［Processor Source Maintenance］メニューが開き、選択したプロセッサの名前が自動的に［Name］フィールドに表示されます。

［Ac］フィールドに「?」（疑問符）を入力すると、ウィンドウが開き、その他の有効なオプションがリストされます。

コマンドプロセッサの名前およびライブラリ名の長さは、1～8 文字です。英大文字（A～Z）、数字（0～9）、および特殊文字（"-"、"/ "、"$"、"&"、"#"、"+"、および "_"）で構成できます。

ヘッダーレコード

ヘッダーメンテナンス機能では、コマンドプロセッサに対するさまざまなグローバル設定を定義します。これらの定義をまとめてヘッダーと呼びます。ヘッダーを作成および修正するために、7つのヘッダーメンテナンス画面が用意されています。コマンドプロセッサのヘッダー設定は、開発の任意の段階で更新できます（以下のセクションを参照）。設定を修正した後、コマンドプロセッサのステータスは常に Header に設定されます（「*Current Status*」も参照）。

以下に参考情報を示します。

- コマンドプロセッサの新規作成
- ヘッダー修正 - 全般的な説明
- キーワードランタイムオプション - ヘッダー 1
- キーワードエディタオプション - ヘッダー 2
- その他のオプション - ヘッダー 3
- コマンドデータ処理 - ヘッダー 4
- ランタイムエラー処理 - ヘッダー 5
- 統計 - ヘッダー 6
- ステータス - ヘッダー 7

コマンドプロセッサの新規作成

手順 66.5. 新しいコマンドプロセッサを作成するには

1 ［Processor Source Maintenance］メニューで、ファンクションコード「**N**」（Create New Processor）、
作成するコマンドプロセッサの名前、および
後でコマンドプロセッサをカタログする Natural ライブラリの名前を入力します。

2 ENTER キーを押します。

最初のヘッダーメンテナンス画面が表示されます。

最初のヘッダーメンテナンス画面および後続の画面には、編集可能なデフォルト値が入力されています。

ヘッダー修正 - 全般的な説明

ヘッダー修正機能を使用して既存のヘッダーをメンテナンスします。つまり、指定したコマンドプロセッサのさまざまなヘッダー設定を修正します。

手順 66.6. 既存のヘッダーを修正するには

1 ［Processor Source Maintenance］メニューで、ファンクションコード「**H**」（Modify Header）、
該当するコマンドプロセッサの名前、および
このコマンドプロセッサがカタログされているライブラリの名前を入力します。

2 ENTER キーを押します。

最初のヘッダーメンテナンス画面が表示されます。

3 以下に示すヘッダーメンテナンス画面の入力フィールドを修正します。

4 ENTER キーを押して修正を確認します。

プロセッサヘッダーの定義とメンテナンスには、7種類の画面を使用できます、ヘッダーの定義については前のセクションを参照してください。

手順 66.7. ヘッダーメンテナンス画面間を移動するには

■ PF8 キー（前へ）または PF7 キー（後へ）を使用します。

各画面には、次の情報が表示されます。

Name	コマンドプロセッサの名前。
ライブラリ	作成したコマンドプロセッサオブジェクトを、カタログ後に保存するライブラリの名前。
DBID、FNR	指定したライブラリが存在するデータベースの ID およびファイル。
Created by	このコマンドプロセッサを初期化した Natural ユーザーのユーザー ID。
日付	コマンドプロセッサが新規作成された日付。
Current Status	コマンドプロセッサのステータス。 Init　コマンドプロセッサが初期化されました。 Header　コマンドプロセッサのヘッダーが作成または修正されました。 Keysave　キーワードが定義および保存されました。 Keystow　キーワードがチェックおよび格納されました。 Function　キーワード結合が定義されました。 Action　ランタイムアクションが定義されました。 Object　コマンドプロセッサのオブジェクト形式が作成されました。 Frozen　コマンドプロセッサがフリーズされました。

	Copied　コマンドプロセッサがコピーされました。 Error　エラーが検出されました。

キーワードランタイムオプション - ヘッダー 1

前述のとおりにヘッダー修正機能を選択すると、［**Processor Header Maintenance 1**］画面が表示されます。

```
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
```

コマンドプロセッサに定義するキーワードに適用されるさまざまな属性をこの画面に入力します。

フィールド	説明
First Entry used as	キーワードシーケンスの定義時に、1 番目のエントリ（エントリタイプ 1）として入力したすべてのキーワードと関連付けられる記述テキスト。 例えば、キーワードシーケンスの 1 番目のキーワードが実行するアクション（DISPLAY、DELETE など）を示す場合、このフィールドに記述テキスト「Action」を入力します。 「*キーワードのメンテナンス*」セクションで説明されているように、このフィールドに入力したテキストの最初の 4 文字は、キーワードエディタの［**Use**］というヘッダー名の列に表示されます。
Second Entry used as	キーワードシーケンスの定義時に、2 番目のエントリ（エントリタイプ 2）として入力したすべてのキーワードと関連付けられる記述テキスト。 例えば、キーワードシーケンスの 2 番目のキーワードが使用するオブジェクト（DOCUMENT、FILE など）を示す場合、このフィールドに記述テキスト「Object」を入力します。 「*キーワードのメンテナンス*」セクションで説明されているように、このフィールドに入力したテキストの最初の 4 文字は、キーワードエディタの［**Use**］というヘッダー名の列に表示されます。
Third Entry used as	キーワードシーケンスの定義時に、3 番目のエントリ（エントリタイプ 3）として入力したすべてのキーワードと関連付けられる記述テキスト（TITLE、PARAGRAPH など）。 「*キーワードのメンテナンス*」セクションで説明されているように、このフィールドに入力したテキストの最初の 4 文字は、キーワードエディタの［**Use**］というヘッダー名の列に表示されます。
Minimum Length	キーワードを定義するときに指定可能な最小文字数。有効な値は 1〜16 文字です。デフォルトは 1 文字です。
Maximum Length	キーワードを定義するときに指定可能な最大文字数。有効な値は 1〜16 文字です。デフォルトは 16 文字です。
Dynamic Length Adjustment	可能な値は次のとおりです。 + ランタイム時に、各キーワードを完全な形で入力する必要があります。 - ランタイム時に、各キーワードの省略形を指定できます。ただし、他のキーワードに対して一意性を保てる場合に限ります。 S 指定したキーワードに必要な入力文字数は、「*キーワードのメンテナンス*」で説明されているように、キーワードの定義時にキーワードエディタの［**ML**］フィールドで指定します。
Keyword Sequence	ランタイム時に処理される、キーワードエントリの順番。指定できる値は1、2、3、および P（パラメータインジケータ用）です。デフォルトは 12 で、最初が 1 番目のキーワードエントリ、次が2番目のキーワードエントリという順番になります。「*キーワードのメンテナンス*」の［**E**］フィールドも参照してください。
Alternative Sequence	デフォルトの順番（上記で指定）がランタイム中にエラーになった場合に、ランタイムで処理されるキーワードの代替順序。

フィールド	説明
Local/Global Sequence	ランタイム時にコマンドを有効にする順序を指定します。可能な値は次のとおりです。 L コマンドをローカルコマンドとして有効にします。 G コマンドをグローバルコマンドとして有効にします。 デフォルトの有効順序は LG です。この指定は、コマンドを最初にローカルコマンドとして有効にし、次に、必要に応じて、グローバルコマンドとして有効にします。

キーワードエディタオプション - ヘッダー 2

［**Processor Header Maintenance 2**］画面では、追加のキーワード属性を入力できます。

フィールド	説明
Header 1 for User Text	この 2 つのフィールドには記述テキストを入力します。このテキストは、キーワードエディタの、ユーザーテキスト用に確保されている列の上に表示されます。またこのテキストは、『ステートメント』ドキュメントの説明に従って PROCESS COMMAND ステートメントで TEXT オプションを指定すると、ランタイム中に出力されます。
Header 2 for User Text	
Prefix Character 1	このフィールドと次の 3 つのフィールドを使用して、キーワードに 16 進接頭辞を付けます。これにより、通常のキーボードでは表せない内部キーワードの処理が可能になります。コマンドプロセッサがカタログされると、キーワードのすべての接頭文字が、指定された 16 進値で置き換えられます。 ［Prefix Character］フィールドのいずれかに空白以外の文字を入力した場合、指定された文字は［Hexadecimal Replacement］フィールドに指定した 16 進値で置き換えられます。
16 進置換 1	このフィールドに指定した値が［Prefix Character］フィールドに指定した文字に代わって、ランタイム時にキーワードの接頭辞として使用されます。
Prefix Character 2	上記の Prefix Character 1 を参照してください。
16 進置換 2	上記の 16 進数を参照してください。置換 1.
Keywords in Upper Case	キーワードエディタおよびアプリケーション内で、キーワードを大文字に変換するかどうかを指定します。 Y キーワードエディタで入力されたキーワードは、自動的に大文字に変換されます。アプリケーションでは、エンドユーザーはキーワードを大文字でも小文字でも入力できます。 N キーワードエディタで入力されたキーワードは、大文字に変換されません。アプリケーションでは、エンドユーザーはキーワードエディタに実際に表示されるとおりにキーワードを入力する必要があります。
Unique Keywords	プロセッサ内のキーワードを一意にする必要があるかどうかを指定します。 Y 各キーワードは、そのタイプに関係なくプロセッサ内で一意に定義する必要があります。

フィールド	説明
	N 各キーワードは、指定のキーワードタイプ（1、2、3、またはP）ごとに一意に定義する必要があります。

その他のオプション - ヘッダー 3

［**Processor Header Maintenance 3**］画面では、その他のオプションを入力できます。

フィールド	説明
Invoke Action Editor	機能エディタからランタイムアクションエディタを呼び出すかどうかを指定します。「*ランタイムアクションエディタ*」および「*機能定義*」を参照してください。 Y 有効なキーワード結合を機能エディタで定義すると、必ずランタイムアクションエディタが呼び出されます。 N 機能エディタでは、ランタイムアクションエディタは抑制されます。 **注意:** Natural システムライブラリ SYSNCP で提供されているユーザー出口 NCP-REDM を使用する場合、このオプションはYに設定する必要があります。Yに設定しないと、不正なランタイムアクションの値をこの時点で検出できず、ランタイムエラーを引き起こす可能性があります。
Catalog User Texts	ユーザーテキストをコマンドプロセッサとともにカタログするかどうかを指定します。 Y コマンドプロセッサをカタログするときに、編集行（キーワードエディタ）（「*キーワード定義*」を参照）のテキスト部分、およびアクション行（ランタイムアクションエディタ）のユーザーテキスト部分が、関連するキーワードまたは機能に結合されます。その後、このテキストは、PROCESS COMMAND ステートメントの TEXT オプションを使用して、ランタイム時に読み込むことができます。 N テキストはコマンドプロセッサとともにカタログされず、ランタイム時に読み込むこともできません。
Security Prefetch	セキュリティチェックを、ランタイム中最初にコマンドプロセッサが呼び出されたときに実行するか、または各コマンドの評価時に実行するかを指定します。 Y Natural Security がインストールされている場合、セキュリティチェックは、コマンドプロセッサが呼び出されたときにすべてのキーワードに対して実行されます。 N Natural Security がインストールされている場合、セキュリティチェックは、各キーワードの評価と一緒に実行されます。 オプションYを選択した場合、セキュリティチェックは、コマンドプロセッサが呼び出されたときにすべてのキーワードに対して1回のみ実行されます。チェック処理は時間がかかるため、ランタイム時の最初のコマンドの評価は比較的時間がかかりますが、残りすべてのコマンドの評価は比較的速く処理されます。逆に、オプションNを選択した場合、セキュリティはコマンドが評価される前にキーワードごとに個別にチェックされるため、各コマンドの評価時間は常に同じとなります。

フィールド	説明
Command Log Size	ランタイム時に処理されたコマンドは、コマンドプロセッサによってコマンドログエリアに保存されます。コマンドログに割り当てるストレージスペースを KB 数で入力フィールドに指定します。 0 コマンドログにストレージスペースを割り当てません。コマンドログは作動しません。 1 コマンドログに 1 KB のストレージスペースを割り当てます。コマンドログは作動します。
Implicit Keyword Entry	後続するすべてのコマンドに対する暗黙的なキーワードとして、タイプ 1 のキーワードを保持するかどうかを指定します。 1 タイプ2のキーワードのみを含むコマンドを入力した場合、コマンドプロセッサによって、最後に入力したタイプ 1 のキーワードが暗黙的なキーワードとみなされます。 N オプションは無効です。
Command Delimiter	コマンド行に複数のコマンドを指定する場合は、コマンドを区切るために使用する文字を指定します。ランタイム時には、先頭のコマンドのみが実行されます。 例えば、次のようになります。 DISPLAY CUSTOMER; MODIFY CUSTOMER; PRINT
PF-Key may be Command	PF キーにコマンドを割り当てることができるかどうかを指定します。ランタイム時にコマンドプロセッサですべて空白のコマンド行を受け取ると、ユーザーによって PF キーが押されたかどうかがチェックされます。 可能な値は次のとおりです。 A この PF キーの ID（システム変数 *PF-NAME）をコマンドとして使用します。 K *PF-KEY システム変数の内容をコマンドとして使用します。 Y *PF-NAME が空の場合、*PF-KEY システム変数の内容を代わりに使用します。 N PF キーはコマンドとして使用できません。「コマンド行を受け入れられません」というメッセージとともに Natural エラー NAT6913 が発行されます。 システム変数 *PF-NAME および *PF-KEY の詳細については、『システム変数』ドキュメントを参照してください。

コマンドデータ処理 - ヘッダー 4

［**Processor Header Maintenance 4**］画面に入力する属性で、機能に対するコマンドデータの処理方法を指定します。コマンドデータはオプションです。

オプションは以下のとおりです。

フィールド	説明
Data Delimiter	データの前に付ける文字を指定します。デフォルトのデータデリミタは "#" です。 例：ADD CUSTOMER #123
Data Allowed	ランタイム時にデータ入力を認めるかどうかを指定します。 N データが見つかると、ランタイムエラーが発生します。 D データが存在する場合には取り除かれます。 S データはNaturalスタックの最上部に表示されます。チェックは実行されません。 Y データはチェックされ、タイプ P（パラメータインジケータ）のキーワードエントリが評価されます。 Y を指定した場合の例：DISPLAY CUSTOMER NAME=SMITH
More than one Item Allowed	適用されるのは、［Data Allowed］オプションがYに設定されている場合に限られます。複数のデータ文字列を許可するかどうかを指定します。 N 複数のデータ文字列が見つかると、ランタイムエラーが発生します。 D 先頭のデータ文字列より後の全データが取り除かれます。 Y 複数のデータ文字列を許可します。 例：ADD ARTICLE #111 #222 一意性が保証される場合に限り、データデリミタを省略できます。 例：ADD ARTICLE 123
Maximum Length of one Item	適用されるのは、［Data Allowed］オプションがYに設定されている場合に限られます。 データ文字列に許可される最大文字数を指定します。指定の最大文字数を超過すると、ランタイムエラーが発生します。有効範囲：1～99。
Item Must be Numeric	適用されるのは、［Data Allowed］オプションがYに設定されている場合に限られます。各データ値を整数値にする必要があるかどうかを指定します。 Y データ入力は正の整数値にする必要があります。これに反すると、ランタイムエラーが発生します。 N データはどのようなタイプでも許可されます。
Put to Top of Stack	適用されるのは、［Data Allowed］オプションがYに設定されている場合に限られます。データを配置する場所を指定します。

フィールド	説明
	Y　データは Natural スタックの最上部に表示されます。 1-9　データは DDM フィールド RESULT-FIELD の *n* 番目のオカレンスに配置されます。ランタイムアクションの結果としてオカレンスにすでに値が入っていた場合は、上書きされます。
If Error, Drop all Data	適用されるのは、［Data Allowed］オプションが Y または N に設定されている場合に限られます。データ評価エラーに対する処理を指定します。 Y　データの評価時にエラーが発生した場合、データは破棄され、処理は続行します。 N　データの評価時にエラーが発生した場合、下記の説明のようにエラーハンドラに制御が渡ります。

ランタイムエラー処理 - ヘッダー 5

［**Processor Header Maintenance 5**］画面に入力する属性で、ランタイムエラーの処理方法を指定します。

フィールド	説明
General Error Program	ランタイム処理中にコマンドプロセッサによってエラーが検出された場合に、制御を受け取るプログラムの名前。このプログラムが呼び出されるとき、Natural スタックには次の情報が格納されています。 エラー番号　（N4） 行番号　（N4） ステータス　（A1） プログラム名　（A8） レベル　（N2） エラープログラムも特定のエラー処理も指定しなかった場合（下記参照）、Natural システム変数 *ERROR-TA に含まれている名前のプログラムが呼び出されます。それ以外の場合には、Natural システムエラーメッセージが発行されます。
Keyword not found	キーワードを見つけられなかった場合に実行するアクションが指定されているかどうかを示します。
Keyword missing	キーワードタイプがなかった場合に実行するアクションが指定されているかどうかを示します。
Keyword Sequence Error	キーワードシーケンスエラーの場合に実行するアクションが指定されているかどうかを示します。
Command not defined	未定義コマンドの場合に実行するアクションが指定されているかどうかを示します。
Data disallowed	禁止されたデータの場合に実行するアクションが指定されているかどうかを示します。

フィールド	説明
Data Format/Length Error	フォーマット／長さエラーの場合に実行するアクションが指定されているかどうかを示します。
General Security Error	全体的なセキュリティチェック中にエラーが検出された場合に実行するアクションが指定されているかどうかを示します。
Keyword Security Error	キーワードのセキュリティチェック中にエラーが検出された場合に実行するアクションが指定されているかどうかを示します。
Command Security Error	コマンドのセキュリティチェック中にエラーが検出された場合に実行するアクションが指定されているかどうかを示します。

統計 - ヘッダー 6

［**Processor Header Maintenance 6**］画面には、コマンドプロセッサに指定したキーワードに関する統計データをレポートする出力フィールドのみが表示されます。

次の統計情報が表示されます。

フィールド	説明
Entry *n* Keywords	コマンドプロセッサに定義されたタイプ *n* のキーワードの数。シノニムは含まれません。
Entry *n* Keywords + Synonyms	タイプ *n* のキーワードとそのシノニムの合計。
Highest IKN for Entry *n*	キーワードタイプ *n* の最大内部キーワード番号。
Possible Combinations	定義されたキーワードの可能な結合の数。
Cataloged Functions	現在カタログされているキーワード結合の数。

ステータス - ヘッダー 7

［**Processor Header Maintenance 7**］画面には、コマンドプロセッサの各部分が実行または修正された日時をレポートする出力フィールドのみが表示されます。

キーワードのメンテナンス

キーワードは、機能を定義するための基本的なコンポーネントです。キーワードを定義する前に、ヘッダーメンテナンスレコードを作成する必要があります。「*ヘッダーレコード*」を参照してください。

- キーワード定義
- エディタコマンド
- 位置調整コマンド

- 行コマンド

キーワード定義

コマンドで使用するキーワードは、キーワード定義機能とキーワードエディタで作成します。キーワードエディタは、エディタ行が別々のフィールドに分割されていること以外は、既存の Natural エディタと類似しています。Natural プログラムエディタで使用するエディタコマンド（関連するセクションを参照）および行コマンド（関連するセクションを参照）の大部分は、キーワードエディタでも使用できます。

手順 66.8. キーワードエディタを呼び出すには

1 ［Processor Source Maintenance］メニューで、ファンクションコード「**K**」（Define Keywords）を入力します。

2 ENTER キーを押します。

キーワードエディタ画面が表示されます。

以下のキーワードエディタ画面が表示されます。このセクションの例として使用するキーワードがすでにいくつか定義されています。

```
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
```

コマンド言語に必要なすべてのキーワードをキーワードエディタに入力します。キーワードは任意の順番で入力できます。ただし、シノニムは関連するキーワードの直後に続ける必要があります。各キーワードに対して、そのキーワードがコマンド構文のどの部分に属するかを指定するタイプを割り当てます。コマンドプロセッサのコマンド構文規則は、プロセッサヘッダーで指定します。「ヘッダーレコード」の「*キーワードランタイムオプション - ヘッダー 1*」を参照してください。例えば、キーワードを、タイプ 1（コマンドの先頭に入力）、タイプ 2、タイプ 3、別のキーワードのシノニム、またはパラメータインジケータのいずれにするかを指定できます。

注意: 現在のところ、どのようなコンピュータにも意味論を理解する能力はないため、コマンド言語には厳密な構文が必要です。そのため、言葉による入力が、コマンド言語でやり取りする唯一の実用的な方法となっています。

上記の例では、キーワード DELETE と DISPLAY がタイプ 1 のキーワードとして定義されています。プロセッサヘッダーで指定したとおり、これらのキーワードはアクションを示します。キーワード DOCUMENT はタイプ 2 のキーワードとして定義されており、オブジェクトを示します。ただし、キーワード FILE はタイプ 1 と 2 の両方として定義されているため、コマンド内の位置に応じてアクションまたはオブジェクトを示します。2つのキーワードタイプを結び付けて DELETE FILE や FILE DOCUMENT などのコマンドを作ることができます。

コマンド行から SAVE または STOW コマンドを発行して、入力したキーワードを保存することができます。STOW コマンドは、キーワード定義をソース形式で保存することに加えて、それらの整合性チェックも実行します。キーワードが正常に格納されると、ランタイム時にコマンドを評価するために使用される内部キーワード番号（IKN）がキーワードに与えられます。シノニムは常にマスタキーワードと関連付けられ、マスタの IKN を取得します。

キーワードエディタの各行には以下のフィールドが含まれています。

フィールド	説明
I	出力フィールド。この情報フィールドには次の値が表示されます。 E 定義エラーが検出されたことを示します。 X 行が X でマークされています。 Y 行が Y でマークされています。 Z 行が X と Y の両方でマークされています。 S この行にスキャン対象値が見つかりました。
行	出力フィールド。エディタの行番号が表示されます。
E	キーワードのエントリタイプ、つまり、キーワードをコマンドに入力する位置を指定します。エントリタイプは、1 番目、2 番目、3 番目、シノニム、またはパラメータインジケータのいずれかです。 例えば、上記のキーワードエディタ画面の例では、キーワード DELETE はエントリタイプ 1、DOCUMENT はエントリタイプ 2 になっています。これらのキーワードを使用すると、DELETE DOCUMENT コマンドを定義できます。

フィールド	説明
	このフィールドに入力できる文字は以下のとおりです。 1 この行に定義したキーワードは、コマンドシーケンスの1番目の項目として使用されます。 2 この行に定義したキーワードは、コマンドシーケンスの2番目の項目として使用されます。 3 この行に定義したキーワードは、コマンドシーケンスの3番目の項目として使用されます。 S この行に定義したキーワードは、前の行に指定したタイプ1、2、3、またはPのキーワードのシノニムとして使用されます。 P この行に定義したキーワードは、コマンドシーケンスでパラメータインジケータとして使用されます。 * この行にはキーワードを定義しません。代わりに、この行はコメント行としてのみ使用されます。 ? この記号は、キーワード指定が誤りであることを示す表示値です。
Use	出力フィールド。このフィールドに表示される値は、［E］フィールドに入力した値で決まります。 1-3 1〜3番目の各キーワードエントリの場合は、プロセッサヘッダーに指定したユーザーテキストの先頭4文字がそれぞれ表示されます。「*ヘッダーレコード*」の「*キーワードエディタオプション - ヘッダー2*」も参照してください。 S シノニムの省略形であるSYNOが表示されます。 P パラメータインジケータの省略形であるPARMが表示されます。
キーワード	定義するキーワードを入力します。空白で埋めることはできません。プロセッサヘッダーで、キーワードに大文字のみ使用できるように指定した場合、キーワードは、どのように入力されたかに関係なく、常に大文字に変換されます。それ以外の場合、大文字／小文字は入力されたままとなります。 キーワードの最大文字数と最小文字数は、ヘッダーに指定した設定で決まります。デフォルトは1〜16文字です。キーワードは、ヘッダーにその他の指定がない限り、一意にする必要があります。「*ヘッダーレコード*」の「*キーワードエディタオプション- ヘッダー2*」で説明されているように、キーワード接頭辞を使用できます。
IKN	出力フィールド。内部キーワード番号（IKN）は、有効な各キーワードに割り当てられる識別子です。IKNは、テストおよびデバッグに役立ちます。IKNは、キーワードが正常に格納されたときにのみ割り当てられます。「*エディタコマンド*」の **STOW** コマンドも参照してください。各キーワードに一意のIKNが割り当てられます。ただし、シノニムにはマスタのIKNが割り当てられます。上記のキーワードエディタ画面の例のDISPLAYおよびSHOWを参照してください。
ML	入出力フィールド。キーワードの最小文字数を示します。「*ヘッダーレコード*」の「*キーワードランタイムオプション - ヘッダー1*」で説明されているように、プロセッサヘッダーの［**Dynamic Length Adjustment**］フィールドに **S** を指定していると、このフィールドは入力フィールドになります。この場合、キーワードに必要な入力文字数を指定する必要があります。S以外の場合は、他のキーワードとの混同を避けるために、ユーザーが指定する必要がある、キーワードの最小文字数が、このフィールドに表示されます。

フィールド	説明
	例えば、上記のキーワードエディタ画面の例の場合、キーワード MENU に必要な入力は **M** のみですが、キーワード DISPLAY は、キーワード DELETE との混同を避けるために、「**DI**」と入力する必要があります。
コメント	キーワードに関するテキストを自由に入力できます。入力に対する制限はありません。「ヘッダーレコード」の「その他のオプション - ヘッダー 3」で説明されているように、ヘッダー定義の［**Catalog User Texts**］フィールドを Y に設定した場合、このユーザーテキストはカタログされたコマンドプロセッサに組み込まれます。このユーザーテキストは、PROCESS COMMAND ステートメントの TEXT オプションを使用して、ランタイム時に読み込むことができます。この列の先頭に表示されるヘッダーテキストは、ヘッダー定義の［Header for User Text 1］および［Header for User Text 2］フィールドで制御されます。

エディタコマンド

キーワードエディタのコマンド行に入力できるコマンドは以下のとおりです。

Command	関数
ADD	エディタの最後の部分に空行を 10 行追加します。
CANCEL	プロセッサメンテナンスメニューに戻ります。
CHECK	キーワードソースの整合性をテストします。
EXIT	プロセッサメンテナンスメニューに戻ります。
HELP	有効なエスケープ文字および他の有効なプロセッサ設定を表示します。
INFO	カーソルが位置付けられているキーワードに関する情報を表示します。
LET	最後に Enter キーを押して以降、現在の画面に行ったすべての修正を無効にします。
POINT	行コマンド **.N** が入力された行を現在の画面の先頭に位置付けます。
RECOVER	最後の SAVE または STOW の前に存在したキーワードソースに戻します。
RESET	現在の X および Y 行マークを削除します。
SAVE	キーワードソースを保存します。
SCAN	スキャンを実行して、スキャン値の次のオカレンスを探します。
STOW	キーワードソースが格納され、有効なキーワードに対して内部キーワード番号（IKN）が生成されます。

位置調整コマンド

エディタの位置付けコマンドは、Naturalプログラムエディタの位置付けコマンドと同じです。詳細については、『エディタ』ドキュメントの「プログラムエディタプログラムエディタ」を参照してください。

エディタの最後の行には、エディタ内の表示位置を示す出力フィールドが含まれています。以下の出力値が表示されます。

Top	エディタは現在キーワードソースの先頭に位置付けられています。
Mid	エディタは現在キーワードソースの中央に位置付けられています。
Bot	エディタは現在キーワードソースの最後に位置付けられています。
Emp	エディタは現在空白です。
すべて	ソース全体が現在の画面に含まれています。

行コマンド

キーワードエディタの行コマンドは、コマンド .J および .S を使用できないこと以外は、Natural プログラムエディタと同じです。

各コマンドは［**E**］フィールドから入力します。コマンドの残りの部分は、次の画面のように、［**Keyword**］フィールドに入力します。

```
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
```

注意: 行を移動（.M）またはコピー（.C）する場合は、個々のキーワードを常にシノニムと一緒に移動またはコピーするようにします。

行を削除（.D）する場合、対応するキーワードおよびこれらのキーワードを含むすべての機能は、STOW エディタコマンドを発行するまで、データベースから削除されません。STOW コマンドを発行しない限り、これらの機能は機能エディタに表示され続けます。

機能のメンテナンス

機能は、キーワードエディタに入力したキーワードで構成されます。機能を定義する前に、キーワードが正常に格納されている必要があります。「*キーワードのメンテナンス*」を参照してください。

- 機能定義
- エディタコマンド
- ダイレクトコマンド QUICK-EDIT
- ローカル機能とグローバル機能
- 機能を有効にする手順

機能定義

機能定義機能および機能エディタを使用して機能を指定し、特定のロケーションからアクセスできる有効なコマンドを構成します。

手順 66.9. 機能エディタを呼び出すには

1 ［Processor Source Maintenance］メニューで、ファンクションコード「**F**」（Define Functions）を入力します。

2 ENTER キーを押します。

機能エディタ画面が表示されます。

機能エディタには、キーワードエディタで格納したキーワードの有効な結合がすべて表示されます。

次の画面は機能エディタを示しています。この画面には、「*キーワードのメンテナンス*」のキーワードエディタ画面で例として使用されているキーワードが含まれています。

```
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
```

アプリケーション内の有効な機能として指定するキーワード結合を、それぞれ有効にする必要があります。キーワード結合は、グローバル機能、ローカル機能、またはその両方として有効にできます。グローバル機能はアプリケーション内のどこからでも呼び出すことができますが、ローカル機能はアプリケーション内の特定のロケーションからのみ呼び出すことができます。

機能エディタ画面の左上隅に表示されている2つのフィールドは、現在有効なモード（ローカルまたはグローバル）および現在キーワード結合を有効にできるロケーションを示しています。上の画面の "Edit Global Combinations" というテキストは、グローバルモードが有効になっていることを示しています。ローカルモードが有効になっている場合は、ここに "Edit Local Combinations" というテキストが表示されます。上の画面では、このテキストの下に "Global" というテキストが表示されています。これは、リストされているすべての結合をグローバルとして有効にできることを示しています。ローカルモードでは、このフィールドに、ローカルとして有効にできるロケーションの名前が表示されます（例："Local DISPLAY FILE"）。

機能エディタには次の列が含まれています。

列	説明
I	出力フィールド。機能編集の結果として次の値が出力されます。 E ランタイムアクションが編集されました。 D 参照されるロケーションが表示されました。 V 有効性が発行されました。 R 有効性が削除されました。
Ac	実行するアクション。次の値を入力できます。 VG グローバル機能として有効にします。 VL ローカル機能として有効にします。 RG グローバル機能としての有効性を削除します。 RL ローカル機能としての有効性を削除します。 DL 指定した機能をローカル機能として参照するすべての機能を表示します。 EG グローバル機能のランタイムアクションエディタを呼び出します。「ランタイムアクション」の「*ランタイムアクションエディタ*」を参照してください。 EL ローカル機能のランタイムアクションエディタを呼び出します。「ランタイムアクション」の「*ランタイムアクションエディタ*」を参照してください。 +G 任意のグローバル機能をメンテナンスできるように、グローバルモードを呼び出します。 +L 現在の行のローカル機能をメンテナンスできるように、この行のローカルモードを呼び出します。 IN この行にキーワードの情報が表示されます。
Action	この 3 つの列を使用して、現在定義されているキーワードの有効な結合をすべて表示します。
オブジェクト	
Addition	各キーワード列の上部に表示されるテキストは、プロセッサヘッダーで指定した［**First Entry used as**］、［**Second Entry used as**］、および［**Third Entry used as**］の各フィールドで制御されます。「ヘッダーレコード」の「*キーワードランタイムオプション - ヘッダー 1*」を参照してください。
Global	機能がグローバルコマンドとして定義されている場合、このフィールドに Yes が表示されます。
Local	機能がローカルコマンドとして定義されている場合、このフィールドには現在のロケーションに対する Yes が表示されます。ただし、ローカルモードの場合のみです。
Any Loc	任意のロケーション。機能がプロセッサ内の別のどこかでローカルコマンドとして定義されている場合、このフィールドには他のいずれかのロケーションに対する Yes が表示されます。

エディタコマンド

機能エディタのコマンド行に入力できるコマンドは以下のとおりです。

Command	関数
ANY ON	［Any Loc］列を使用できるようにします。
ANY OFF	［Any Loc］列を使用できないようにします。列は疑問符で埋められます。これにより、機能エディタ内のスクロールが速くなります。さらに、3 番目の再位置付けフィールドを使用できます。進行中の処理情報ウィンドウは表示されません。
FIELD	キーワード特定の結合を表示します。
GLOBAL	グローバルモードを有効にします。
LOC	次のロケーショングループに位置付けます。
LOC+	1 ロケーション先に位置付けます。
SINGLE ON	単一ワード機能のみを表示します。
SINGLE OFF	使用可能なすべての結合を表示します。
TOP	リストの先頭に位置付けます。

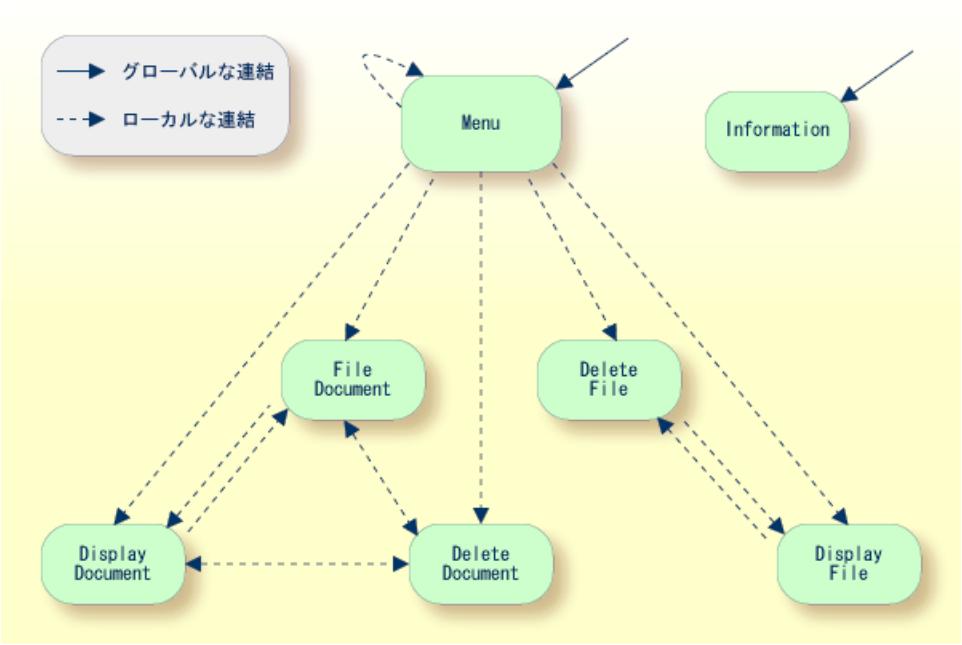
ダイレクトコマンド QUICK-EDIT

ダイレクトコマンド QUICK-EDIT を使用すると、キーワードまたは IKN を直接入力することで、ローカル機能およびグローバル機能と、対応するランタイムアクションをすばやく定義することができます。これは、非常に大きいコマンドプロセッサに役立ちます。ただし、コマンドを発行できるロケーションが検証されないため、ランタイム時にナビゲーションが正しく機能しない場合があることに注意してください。

ローカル機能とグローバル機能

ローカル機能とグローバル機能の概念を理解するには、有効な各キーワードの結合をアプリケーション内のロケーションとして図に示す必要があります。例えば、Display File という名前のロケーションとして示します。機能エディタでは、このロケーションから発行できるコマンドと、DISPLAY FILE コマンドを使用することでこのロケーションに到達できるロケーションを指定します。

サンプルアプリケーション内のローカルおよびグローバルな連結：



上記のサンプルアプリケーションでは、Menu ロケーションと Information ロケーションのみが、グローバルとして指定されています。そのため、これらのロケーションは、サンプルアプリケーション内の残りのすべてのロケーションから直接アクセスできます。Information を除く全ロケーションは、Menu ロケーションに対してローカルとして指定されています。Display File ロケーションから Display Document ロケーションに到達するには、Menu を経由する方法しかありません。

機能を有効にする手順

機能エディタは、グローバルモードとローカルモードの2つのモードで動作します。グローバルモードではグローバル機能を有効にし、ローカルモードではグローバル機能とローカル機能を有効にすることができます。グローバルモードがデフォルトのモードです。エディタの［**I**］フィールドの上にある出力フィールドで、エディタがグローバルモードであるかローカルモードであるかを確認できます。エディタがグローバルモードの場合は、Global と表示されます。エディタがローカルモードの場合には、ローカル機能が有効になるロケーションが表示されます。アプリケーションに対してグローバル機能とローカル機能を有効にする一般的な手順は、次のとおりです。

手順 66.10. グローバル機能とローカル機能を有効にするには

1 グローバルモードの機能エディタで、該当するアクションの隣の［Ac］フィールドに「**VG**」（グローバルを有効にする）を入力して、すべてのグローバル機能を有効にします。

ENTER キーを押します。

［**Runtime Action Definition**］画面が表示されます。

2 PF3 キーを押して、機能エディタに戻ります。

`Yes`がヘッダー名［Global］列の下、検証された機能の横に表示されます。

3 前の手順で有効にした各グローバル機能の［Ac］フィールドに「**+L**」を入力して、ローカルモードに切り換えます。

ENTER キーを押します。

4 このグローバル機能のロケーションとして使用する各機能の［**Ac**］フィールドに、「**VL**」（ローカルを有効にする）を入力します。

ENTER キーを押します。

［Runtime Action Definition］画面が表示されます。

5 PF3 キーを押して、機能エディタに戻ります。

`Yes` がヘッダー名［Local］列の下、検証された機能の横に表示されます。

6 *ローカル*ロケーションに対してローカル機能を有効にするには：前の手順で有効にした各ロケーションの［**Ac**］フィールドに「**+L**」（ローカルモードを呼び出す）を入力して、このロケーションから使用するすべてのローカル機能を有効にします。

ENTER キーを押します。

7 現在のロケーションに対するローカル機能として使用する各機能の［**Ac**］フィールドに、「**VL**」（ローカルを有効にする）を入力します。

8 PF3 キーを押して、機能エディタに戻ります。

`Yes` がヘッダー名［Local］列の下、検証された機能の横に表示されます。

注意: コマンドプロセッサヘッダー（［Processor Header Maintenance 3］）の［Invoke Action Editor］フィールドを `Y` に設定していると、［Runtime Action Definition］ウィンドウがアクションごとに表示されます。「ランタイムアクション」の「*ランタイムアクションエディタ*」を参照してください。

ランタイムアクション

機能エディタで有効なキーワード結合をローカル機能またはグローバル機能として特定した後、各機能を1つ以上のランタイムアクションと関連付けることができます。ランタイムアクションは、機能が発行されると常に実行される 1 つ以上の手順で構成されます。

以下に参考情報を示します。

- ランタイムアクション定義

- ランタイムアクションエディタ

ランタイムアクション定義

SYSNCP には、ランタイムアクションを定義できる場所が 2 つあります。機能エディタ（「*機能のメンテナンス*」を参照）と結果エディタです。このセクションでは、結果エディタについて、機能に対するランタイムアクションの指定方法を説明します。

手順 66.11. 結果エディタを呼び出すには

1 ［**Processor Source Maintenance**］メニューで、ファンクションコード「**R**」（Define Runtime Actions）を入力します。

2 ENTER キーを押します。

結果エディタ画面が表示されます。

```
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
```

結果エディタには、機能エディタで指定したすべてのローカル機能とグローバル機能が表示されます。エディタの各行には、コマンドを発行できるロケーション（［Location］フィールド）、コマンド（［Command］フィールド）、およびコマンドが発行されたときに実行されるアクションの短縮形（［Result］フィールド）が表示されます。

次の表は画面フィールドの説明です。

フィールド	説明
I	出力フィールド。この行で最後に実行されたアクションについての情報。
Ac	実行するアクション。次の値を入力できます。 DI この機能に対するランタイムアクション定義を表示します。 ED この機能に対するランタイムアクション定義を編集します。 PU この機能を削除します。
場所	出力フィールド。 コマンド（下記の［Command］フィールドを参照）を発行できるアプリケーション内のロケーション。機能がグローバルの場合、このフィールドに < Global> が表示されます。コマンドは任意のロケーションから発行できます。
Command	出力フィールド。コマンド。 長いキーワードを使用しているときには、［Location］および［Command］フィールドの内容が切り捨てられる場合もあります。
Result	出力フィールド。 コマンドが発行されたときに実行されるアクションの省略形が表示されます。先頭文字は **Keep Location** 情報（下記参照）を示します。他の文字については、下記のランタイムアクション定義の表を参照してください。

ランタイムアクションエディタ

ランタイムアクションエディタを使用して、特定のロケーションからコマンドが発行されたときに実行するアクションを定義します。ランタイムアクションエディタは、グローバルまたはローカル機能として定義された機能に対してのみ呼び出されます。このエディタは、機能エディタまたは結果エディタから呼び出すことができます。

手順 66.12. 機能エディタからランタイムアクションエディタを呼び出すには

1 グローバル機能の場合は、［**Ac**］フィールドに「**EG**」（グローバル編集）を入力します。

または:

ローカル機能の場合は、［**Ac**］フィールドに「**EL**」（ローカル編集）を入力します。

2 ENTER キーを押します。

手順 66.13. 結果エディタからランタイムアクションエディタを呼び出すには

1 ［**Ac**］フィールドに、「**ED**」を入力します。

2 ENTER キーを押します。

［**Runtime Action Definition**］ウィンドウが表示されます。

```
                         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_____________________________________________________
_  ______________________________________________________________
_  ______________________________________________________________
_  ______________________________________________________________
_  ______________________________________________________________
_  ______________________________________________________________
_  ______________________________________________________________
_  ______________________________________________________________
```

アクションは常に発生元と宛先に関連付けられます。発生元はコマンドが発行されるロケーションであり、宛先はコマンド自体です。このように、コマンドが使用される関係を基準にして、1つのコマンドに異なるアクションを関連付けることができます。

ランタイムアクションエディタでは、アクションが実行された後ロケーションをそのままにするか、またはコマンド自体を新たな現行ロケーションにするかも指定します。

アクションは、エディタの左端の列に1文字コードを入力して指定します。コードの隣のフィールドに、アクションに付随するすべてのパラメータを入力します。このフィールドに「/*」を入力すると、後続の入力はすべてコメントとみなされます。必要なパラメータを省略すると、入力を求めるメッセージが表示されます。

ランタイム時に実行されるアクションの順番は、エディタのエントリ順序（上から下へ）で決定されます。そのため、FETCH が指定されると、その下に指定されたいずれのアクションも実行されません。

ランタイムアクションエディタのフィールドは次のとおりです。

フィールド	説明
場所	出力フィールド。コマンドを発行するロケーション。機能がグローバルとして定義されている場合、<Global> が表示されます。
Command	出力フィールド。このコマンドに対するアクションを指定します。
Keep Location	アクションが実行された後、現在のロケーションをアクティブにするか、または新しいロケーションをアクティブにするかを指定します。このフィールドの値は、EXEC オプションを指定した場合にのみコマンドに影響します。可能な値は次のとおりです。 K 現在のロケーションを保持します。実行するアクションは、現在のロケーションにのみ影響します。 S 新しいグローバルまたはローカルロケーションを設定します。アクションを実行すると、コマンドプロセッサはそのコマンドを新たな現行ロケーションにします。この後に入力するコマンドはすべて、この新ロケーションのローカルコマンドまたはグローバルコマンドのいずれかである必要があります。 **注意:** 定義したアクション自体は、ロケーションに影響を与えません。つまり、実行するアクションは現在のロケーションを変更しません。
Other Options	その他のオプションはすべて、このコマンドシーケンスで提供されるパラメータの処理と関連します。詳細については、「ヘッダーレコード」の「*コマンドデータ処理- ヘッダー 4*」を参照してください。 これらのオプションのヘッダーデフォルトを有効にするには、「*」（アスタリスク）を入力します。

手順 66.14. ランタイムアクションを定義するには

1 前述のように、［**Runtime Action Definition**］ウィンドウを呼び出します。

2 フィールド［**A**］にアクションコードを入力し、対応するフィールドにアクションを入力します。

Code	ランタイムアクション定義
V	デフォルト値です。ランタイムアクションを指定しません。
T	PROCESS COMMAND ステートメントの TEXT または GET オプションを使用して、ランタイム時に読み込めるテキストです。
M	コマンド行を修正します。このデータはコマンド行に置かれます。
C	コマンド。このコマンドは Natural スタックの先頭に置かれます。アスタリスク（*）を指定した場合、この PROCESS COMMAND ステートメントを発行したプログラムの名前がスタックの先頭に置かれます（STACK TOP COMMAND '*PROGRAM'）。(*)
D	データ。このデータは Natural スタックの先頭に置かれます。(*)
F	Natural プログラム名。このプログラムに対して FETCH ステートメントが実行されます。(*)
S	Natural の STOP ステートメント。ランタイム時にこのステートメントが実行されます。(*)
E	この行に指定した値は、即座に Natural システム変数 *ERROR-NR に転送されます。

Code	ランタイムアクション定義
R	DDM フィールド RETURN-CODE にリターンコードが入力されます。『ステートメント』ドキュメントの PROCESS COMMAND を参照してください。
1～9	テキスト文字列。この値は複数の DDM フィールド RESULT-FIELD に入力されます。『ステートメント』ドキュメントの PROCESS COMMAND を参照してください。
*	コメント行。

* これらのアクションは、PROCESS COMMAND ステートメントの EXEC オプションでのみ実行されます。

3 PF3 キーを押して、［**Runtime Action Definition**］ウィンドウを閉じます。

注意: ユーザー出口 NCP-REAM を使用すると、上記のコードの一部またはすべてを使用できます。ユーザー出口 NCP-REEM を使用すると、ランタイムアクション定義テーブルのヘッダーの下の行を修正できます。ユーザー出口 NCP-REDM を使用すると、ランタイムアクション定義のデフォルト値を定義できます。このユーザー出口を使用する場合は、「ヘッダーレコード」の ***Invoke Action Editor*** も参照してください。上記のユーザー出口はすべて Natural システムライブラリ SYSNCP で提供されています。

プロセッサのカタログ化

コマンドプロセッサで使用するすべての機能にランタイムアクションを指定した後、コマンドプロセッサをカタログする必要があります。コマンドプロセッサをカタログすると、プロセッサタイプの Natural オブジェクトが生成されます。

手順 66.15. コマンドプロセッサをカタログするには

1 ［Processor Maintenance］メニューで、ファンクションコード「**C**」（Catalog Processor）、カタログするコマンドプロセッサの名前、
およびコマンドプロセッサをカタログする Natural ライブラリの名前を入力します。

2 ENTER キーを押します。

注意: Natural Security がインストールされている場合、コマンドプロセッサの使用を許可する必要があります。『*Natural Security*』ドキュメントの「*Functional Security*」を参照してください。

Windows、**UNIX**、および **OpenVMS** に関する注意事項：
メインフレームとは異なり、コマンドプロセッサをカタログしても SYSNCP によるレポートは作成されません。

管理者サービス

SYSNCP には、コマンドプロセッサの管理機能が用意されています。*Natural Security* で定義されたシステム管理者のみが、これらのサービスにアクセスする権限を持ちます。

手順 66.16. 管理者サービスにアクセスするには

1 ［**Processor Source Maintenance**］メニューで、ファンクションコード「**A**」（管理者サービス）を入力します。

2 ENTER キーを押します。

［Administrator Services］画面が表示されます。

```
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
```

注意: Natural Security がインストールされていない場合、すべてのユーザーが管理者ステータスになることに注意してください。

以下に参考情報を示します。

- プロセッサ選択
- プロセッサソースのコピー

- プロセッサソースの削除
- ソース／オブジェクト／NCP バッファのプリント
- プロセッサのアンロード
- プロセッサのロード
- プロセッサソースのフリーズ
- Natural Security からの参照

プロセッサ選択

「*プロセッサの選択*」を参照してください。

プロセッサソースのコピー

プロセッサソースのコピーでは、プロセッサ全体をコピーするか、または選択したソース（ヘッダー、キーワード、機能、ランタイムアクション定義）のみをコピーするかを選択できます。

手順 66.17. コマンドプロセッサをコピーするには

1 ［Administrator Services］メニューで、ファンクションコード「**C**」を入力します。

2 ENTER キーを押します。

［Copy Processor Source］ウィンドウが開き、ソースおよびターゲット情報が表示されます。

```
                         Copy Processor Source

                Source           Target

Name ........ SAGTEST_         ________
Library ..... SYSNCP__         SYSNCP__
DBID ........ 10___            10___
FNR ......... 32___            32___
Password ....
Cipher Key ..

Replace ..... NO_
```

3 ［Source］側のフィールドに、コピーするプロセッサの名前、プロセッサが保存されているライブラリ、データベース ID（DBID）、およびファイル番号（FNR）を入力します。デフォルト値は［**Administrator Services**］メニューで指定したプロセッサです。

［**Target**］側のフィールドには、コピー先のプロセッサの名前、コピーしたプロセッサを保存するライブラリ、データベース ID（DBID）、およびファイル番号（FNR）を入力します。

ソースファイルやターゲットファイルがパスワードとサイファキーの両方またはいずれかで保護されている場合には、［**Cipher Key**］フィールドに適切なパスワードとサイファキーの両方またはいずれかを入力します。

ターゲット環境のプロセッサを上書きする場合は、［**Replace**］フィールドに「YES」を入力します。このフィールドのデフォルトは NO です。

4 ENTER キーを押します。

ソースを選択するための次のようなウィンドウが表示されます。

```
                          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 該当する［**Mark**］フィールドに任意の文字を入力して、コピーするソースを選択します。

6 ENTER キーを押します。

プロセッサソースの削除

この機能は、プロセッサソースを削除するために使用します。

手順 66.18. コマンドプロセッサを削除するには

1 ［**Administrator Services**］メニューで、ファンクションコード「**D**」を入力します。

2 ENTER キーを押します。

［**Delete Processor Source**］ウィンドウが表示されます。

3 削除するプロセッサの名前、プロセッサが保存されているライブラリ、データベース ID、およびファイル番号を指定します。ファイルがパスワードとサイファキーの両方またはいずれかによって保護されている場合、適切なパスワードとサイファキーの両方またはいずれかを入力する必要があります。

4 ENTER キーを押します。

削除するソースを選択するための次のようなウィンドウが表示されます。

```
                    Delete Processor Source

Mark  Delete                           Available
----  ----------------------------  ---------
 _    Header ......................    yes
 _    Keywords ....................    yes
 _    Functions ...................    yes
 _    Runtime Action Definitions ..    yes


Name SAGTEST   Library SYSNCP    DBID 10      FNR 32
```

各プロセッサソース（ヘッダー、キーワード、機能、ランタイムアクション定義）の右側にあるフィールドは、ソースが存在するかどうかを示しています。コマンドプロセッサの作成は累積的な処理であるため、あるソースを削除する場合、そのソースをベースにしているソースもすべて削除する必要があります。例えば、上の画面で、機能のソースを削除する場合は、ランタイムアクション定義のソースも削除する必要があります。

5 ［**Available**］と示されている各ソースを選択するには、該当する［**Mark**］フィールドに任意の文字を入力します。

6 ENTER キーを押します。

ソース／オブジェクト／NCP バッファのプリント

プロセッサソースの他に、プロセッサオブジェクトおよびNCPバッファもプリントできます。

手順 66.19. コマンドプロセッサ項目をプリントするには

1 **Administrator Services** メニューで、ファンクションコード「**P**」を入力します。

2 ENTER キーを押します。

［**Print Source/Object/NCP-Buffer**］ウィンドウが表示されます。

3 プリントするプロセッサの名前、プロセッサが保存されているライブラリ、データベース ID、およびファイル番号を指定します。ファイルがパスワードとサイファキーの両方またはいずれかによって保護されている場合、適切なパスワードとサイファキーの両方またはいずれかを入力する必要があります。

4 ENTER キーを押します。

5 プリントする項目を選択するための次のようなウィンドウが表示されます。

```
                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
```

各プロセッサソース（ヘッダー、キーワード、機能、ランタイムアクション定義）の右側にあるフィールドは、項目が存在するかどうかを示しています。

［**Printer**］フィールドに入力できる値は、論理プリンタ ID、VIDEO、または SOURCE です。『ステートメント』ドキュメントの DEFINE PRINTER も参照してください。

6 該当する［**Mark**］フィールドに任意の文字を入力してプリントする項目を選択し、［Printer］フィールドに論理プリンタ名、あるいは「VIDEO」または「SOURCE」を入力します。

7 ENTER キーを押します。

プロセッサのアンロード

手順 66.20. コマンドプロセッサをアンロードするには

1 ［**Administrator Services**］メニューで、ファンクションコード「**U**」を入力します。

2 ENTER キーを押します。［**Unload Processor to Work File 3**］ウィンドウが表示されます。

```
                    Unload Processor to Work File 3

                 Source            Target

Name ........ SAGTEST_
Library ..... SYSNCP__          SYSNCP__
DBID ........ 10___
FNR ......... 32___
Password ....
Cipher Key ..

Report ...... NO_
```

3 ［**Source**］側のフィールドに、アンロードするプロセッサの名前、プロセッサが存在するライブラリ、データベース ID、およびファイル番号を入力します。デフォルト値は［**Administrator Services**］メニューで指定したプロセッサです。ファイルがパスワードとサイファキーの両方またはいずれかで保護されている場合は、適切なパスワードとサイファキーの両方またはいずれかを入力します。

4 レポートを出力する場合は、［**Report**］フィールドに「YES」を入力します。デフォルトは NO です。ファイル拡張子を使用する必要はありません。拡張子を使用する場合は、ファイル拡張子 ".sag" を使用する必要があります。

5 ENTER キーを押します。

プロセッサをアンロードすると、すべてのプロセッサソース（ヘッダー、キーワード、機能、ランタイムアクション定義）がワークファイル 3 に書き込まれます。

注意: あるハードウェアプラットフォームから別のハードウェアプラットフォームにコマンドプロセッサを転送するには、オブジェクトハンドラを使用します。

プロセッサのロード

手順 66.21. コマンドプロセッサをロードするには

1 ［**Administrator Services**］メニューで、ファンクションコード「**L**」を入力します。

2 ENTER キーを押します。

プロセッサをワークファイル 3 から Natural ライブラリにロードするための［**Load Processor from Work File 3**］ウィンドウが表示されます。

```
        Load Processor from Work File 3

  Replace existing processors .. N
  Produce load report .......... NO_
```

3 ロードするプロセッサで同じ名前を持つ既存のプロセッサを置き換えるかどうかを、［**Replace existing processors**］フィールドに「**Y**」または「**N**」（デフォルトは N）を入力して指定します。

4 レポートを出力する場合は、［**Produce load report**］フィールドに「YES」(デフォルトは NO)を入力します。

5 ENTER キーを押します。

注意: プロセッサ名とプロセッサのロード先のライブラリはワークファイルから取得されます。

プロセッサソースのフリーズ

プロセッサを現在の状態でフリーズして、ユーザーがこれ以上修正できないようにすることができます。

手順 66.22. コマンドプロセッサをフリーズするには

1 ［**Administrator Services**］メニューで、ファンクションコード「**F**」を入力します。

2 ENTER キーを押します。［**Freeze Processor Source**］ウィンドウが表示されます。

3 フリーズするプロセッサの名前、プロセッサが保存されているライブラリ、データベース ID、およびファイル番号を指定します。ファイルがパスワードとサイファキーの両方またはいずれかによって保護されている場合、適切なパスワードとサイファキーの両方またはいずれかを入力する必要があります。

4 ENTER キーを押します。

5 次に表示されるウィンドウで、プロセッサソースの修正を許可するかまたは禁止するかを **Y** または **N** で指定します。デフォルトは **Y** です。

6 ENTER キーを押します。

Natural Security からの参照

この機能は、Natural Security が稼働している環境でのみ有効です。この機能を使用して、Natural Security から機能のセキュリティの参照を削除します。

Natural Security でプロセッサに対する機能のセキュリティを定義すると、自動的に参照が作成されます。これらの参照はプロセッサソースとともに、FSEC ではなく、FNAT/FUSER システムファイルに保存されます。

手順 66.23. Natural Security 機能から参照を呼び出すには

1 ［**Administrator Services**］メニューで、ファンクションコード「**R**」を入力します。

2 ENTER キーを押します。

［**Delete References**］ウィンドウが表示されます。

3 プロセッサの名前、プロセッサが保存されているライブラリ、データベース ID、およびファイル番号を指定します。ファイルがパスワードとサイファキーの両方またはいずれかによって保護されている場合、適切なパスワードとサイファキーの両方またはいずれかを入力する必要があります。

4 ENTER キーを押します。

5 次に表示されるウィンドウで、主参照、機能参照、および補助参照を削除できます。

コマンドプロセッサの機能のセキュリティの詳細については、『*Natural Security*』ドキュメントの「*Functional Security*」を参照してください。

セッションプロファイル

セッションプロファイルとは、ユーザー定義が可能なデフォルトの集合のことです。セッションプロファイルによって、SYSNCP 画面の表示方法または入力に対する SYSNCP の応答方法が決定されます。セッションプロファイルでは、例えば、セッションに対するデフォルトのコマンドプロセッサ、または画面属性に割り当てる色を定義できます。SYSNCP には、すべての新規ユーザーに発行される STANDARD という標準セッションプロファイルがあります。必要に応じて、複数の異なるセッションプロファイルを作成し、有効にすることができます。

SYSNCP の管理者は、SYSNCP 内のどのセッションプロファイルでもアクセスおよび修正できます。その他のユーザーもすべてのセッションプロファイルにアクセスできますが、修正できるのは自分のユーザー ID で作成されたセッションプロファイル、または自分のユーザー ID と同じ名前をもつセッションプロファイルのみです。

手順 66.24. セッションプロファイルを定義または修正するには

■ ［**Processor Source Maintenance**］メニューのコマンド行から PROFILE コマンドを発行します。

3 つのセッションプロファイルメンテナンス画面のうちの最初の画面が表示されます。

以下に参考情報を示します。

- セッションプロファイル名
- セッションパラメータ - プロファイル 1
- カラー属性 - プロファイル 2
- その他の属性 - プロファイル 3

セッションプロファイル名

標準プロファイル STANDARD またはシステム変数 *USER の値が、デフォルトのプロファイル名として使用されます。

新規セッションプロファイルを定義するとき、パラメータおよび属性はデフォルト値になっています。必要に応じてこれらのデフォルト値を修正し、新しい名前を入力して PF5 キーを押すことで、これらの値を保存できます。

各プロファイル画面の［Session Profile Name］フィールドは、入出力フィールドになっています。そのため、［Profile Name］フィールドに別のプロファイル名を入力して PF5 キーまたは PF4 キーをそれぞれ押すことで、これらのどの画面からでもそのプロファイルを定義、読み込み、または保存することができます。

セッションパラメータ - プロファイル 1

最初のプロファイルメンテナンス画面では、以下のフィールドを修正できます。

フィールド	説明
Apply Terminal Control 1	起動時に SYSNCP によって発行される SET CONTROL ステートメントのパラメータを入力します。 例えば、このフィールドのいずれかに「**Z**」を入力すると、SYSNCP により SET CONTROL 'Z' ステートメントが発行されます。
Apply Terminal Control 2	
Default Processor Name	このセッションで使用するデフォルトのコマンドプロセッサ名。
Default Processor Library	コマンドプロセッサを保存するために使用する Natural ライブラリ。
Cancel Reaction	要求された修正を完了しないで CANCEL コマンドを発行した場合に必ず警告を発行するかどうかを指定します。 W 警告を発行します。 B 警告を発行しないでバックアウトし、キャンセルします。

フィールド	説明
Clear Key Allowed	クリアキーの使用を許可するかどうかを指定します。 N クリアキーの使用を禁止します。 Y クリアキーは有効で、CANCEL と同じ効果を持ちます。
Default Cursor Position	カーソル位置を指定します。 1 カーソルを画面の最初のフィールドに位置付けます。 C カーソルをコマンド行に位置付けます。
Exec/Display Last Command	LAST コマンドで実行される処理を指定します。 E コマンド行で発行した最後のコマンドを実行します。 D コマンド行で発行した最後のコマンドを表示します。

カラー属性 - プロファイル 2

2番目のプロファイルメンテナンス画面では、さまざまな画面属性に色を割り当てることや、既存の色の割り当てを上書きすることができます。

以下のカラーコードを指定して、色を割り当てます。

Code	色
BL	青
GR	緑
NE	デフォルト色
PI	ピンク
RE	赤
TU	空色
YE	黄色

画面属性に対する色の割り当てについては、『*端末コマンド*』ドキュメントの端末コマンド %= も参照してください。

その他の属性 - プロファイル 3

3 番目のプロファイルメンテナンス画面で指定できる属性は次のとおりです。

フィールド	説明
Message Line Position	メッセージを表示する行。推奨値は21です。詳細については、『*端末コマンド*』ドキュメントの端末コマンド %M も参照してください。
Text for PF5 Key	PF5 ファンクションキーは、グローバルな用途（セッション単位）のために予約されています。PF キー行に表示される PF5 キーのテキストをこのフィールドに入力できます。
Command for PF5 Key	PF5 ファンクションキーは、グローバルな用途（セッション単位）のために予約されています。PF5 キーを押したときに実行されるコマンドを、このフィールドに入力できます。

また、画面には、このプロファイルを最後に修正した日付およびユーザーも表示されます。

XI SYSPCI ユーティリティ - 製品の設定および初期化

67 SYSPCI ユーティリティ - 製品の設定および初期化

- 画面を使用したインストール済み製品の設定 438
- ダイレクトコマンドデータを使用した SYSPCI ユーティリティの呼び出し 444

SYSPCI ユーティリティは、Software AG Installer を使用する、Natural またはそのアドオン製品を初めてインストールした後に使用されます。このユーティリティは、環境に応じて、多数のファイル、パラメータ、および個々の設定をセットアップします。

重要: SYSPCI ユーティリティを使用する場合は、ローカルコンフィグレーションファイルで Natural 管理者として定義されている必要があります（『*コンフィグレーションユーティリティ*』ドキュメントの「*管理者割り当て*」も参照）。Natural 管理者でなくても SYSPCI ユーティリティを呼び出すことができますが、選択した製品の設定を開始すると、SYSPCI ユーティリティが初期化ファイルをロードできず、Natural スタートアップエラー 12 が発生します。

SYSPCI ユーティリティを使用すると、次の操作を実行できます。

- 製品に必要な Adabas ファイルに必要な情報を入力します。ファイルが存在しない場合は追加します。
- 必要な Adabas ファイルのデータベース ID を Natural のグローバルコンフィグレーションファイルに入力します。
- 新規または既存の Adabas ファイルのデータベース ID とファイル番号を、製品のデフォルトパラメータファイルに入力します。
- 製品を初期化します。
- オプションで、選択した製品に応じて追加の機能を実行します（製品データのロードなど）。

注意: アップデートのインストール後、必要な Adabas ファイルと必要なデフォルトパラメータファイル内のパラメータが以前にセットアップされている場合は（例えば、初回インストール後）、SYSPCI ユーティリティを起動する必要はありません。以前に設定したパラメータは、更新インストールで保持されます。

次のトピックで説明するように、SYSPCI ユーティリティはさまざまな方法で呼び出すことができます。

画面を使用したインストール済み製品の設定

SYSPCI ユーティリティが環境で検出できるインストール済み Software AG 製品を設定できます。

以下の説明では、SYSPCI ユーティリティの使用方法に関する一般的な情報と、通常ではすべての製品で使用できるオプションについて説明しています。特定の製品用にセットアップする必要があるファイルの詳細については、その製品のインストールドキュメントを参照してください。

手順 67.1. インストール済み製品を設定するには

1 次のコマンドを入力します。

```
SYSPCI
```

注意: Natural Security で保護されている環境で SYSPCI ユーティリティを呼び出すと、Natural Security は SYSPCI 用のユーティリティプロファイルを検証します。

環境で検出されたインストール済み Software AG 製品をリストする画面が表示されます。例えば、次のようになります。

```
13:24:34                ***** NATURAL SYSPCI UTILITY *****            2014-06-26
User SAG           - Product Configuration and Initialization -


                        Select a single product:
                        _  Natural Development Server (NDV)
                        _  Natural Command Processor (NCP)
                        _  Predict (PRD)
                        _  Natural Business Services (NBS)
                        _  System Automation Tools (SAT)
                        _  Entire Output Management (NOM)
                        _  Entire Operations (NOP)
                        _  Natural Security Log (NSL)
                        _  Natural Security (NSC)
                        _  Exit


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

2 設定する製品にマークを付け、ENTER キーを押します。

重要: 画面にリストされているのと同じ順序で製品を設定することをお勧めします。

表示される画面の内容は、選択した製品によって異なります。例えば、次のようになります。

```
10:51:36              ***** NATURAL SYSPCI UTILITY *****           2014-06-16
User SAG          - Product Configuration and Initialization -

     Product selected: Natural Development Server (NDV)

     Mark actions:
       Create new Adabas file .. X
       Use existing Adabas file  _
       Initialize product ...... X                                          ↩


     Adabas file definitions for for the NDV FDIC file:
       Database ID _____  Select _
       File number _____  Select _   File name _______________









     Start selected action(s) .. _  (Y/N)


Enter-PF1---PF2---PF3---PF4---PF5---PF6---PF7---PF8---PF9---PF10--PF11--PF12---
      Help  Files Exit  Check DBIDs FNRs                                Canc  ↩
 ↩
```

3 ［**Use existing Adabas file**］をマークして PF2 キーを押すと、新しい画面が表示されます。

表示される画面の内容は、選択した製品によって異なります。例えば、次のようになります。

```
08:53:16              ***** NATURAL SYSPCI UTILITY *****           2015-07-08
User SAG          - Product Configuration and Initialization -                ↩
                                                                              ↩
                                                                              ↩
                   Select a single entry for product NDV                      ↩
                                                                              ↩
                       _ List FDU file for NDV-FDIC
                       _ List NDVPARM (parameter file)
                       _ Select a parameter file
                                                                              ↩
                                                                              ↩
                                                                              ↩
                                                                              ↩
                                                                              ↩
                                                                              ↩
                                                                              ↩
                       _  Exit                                                ↩
                                                                              ↩
                                                                              ↩
                                                                              ↩
                                                                              ↩
                                                                              ↩
Enter-PF1---PF2---PF3---PF4---PF5---PF6---PF7---PF8---PF9---PF10--PF11--PF12---
      Help        Exit                                                Canc  ↩
  ↩
```

次の項目を選択できます。

オプション	説明
List FDU file for *ppp-nnnn*	このオプションでは、単一製品ファイルの FDU ファイルがリストされます。この場合の *ppp* は製品コード（上の画面の NDV）、*nnnn* はファイル名（上の画面の FDIC）を表します。
List *ppp*PARM (parameter file)	このオプションでは、製品のデフォルトパラメータファイルがリストされます。この場合の *ppp* は製品コードを表します（上の画面の NDV）。
Select a parameter file	このオプションでは、既存のパラメータファイルをリストし、そこから選択できます。

リストされているすべてのファイルについて、`EDIT` コマンドを使用するか PF11 キーを押して編集モードに切り替えることで、ファイルの内容を変更できます。`COMMANDS` コマンドを使用するか PF10 キーを押して、使用可能なコマンドを表示します。

4 選択した製品に関する次の情報を指定します。

オプション	説明
新規 Adabas ファイルを作成します、または既存の Adabas ファイルを使用します	［**Create new Adabas file**］がマークされている場合、最初のファイルは存在してはならず、新しい Adabas ファイルが作成されます。選択した製品によっては複数のファイルが表示されることがあります。すでに他のファイルが 1 つまたは複数存在する場合は、それらのファイルを製品に使用できます。 ［**Use existing Adabas file**］がマークされている場合、最初のファイルが存在している必要があります。選択した製品によっては複数のファイルが表示されることがあります。他のファイルが存在しない場合は、該当するファイルが作成されることがあります。 ただし Adabas ファイルがすでに存在する場合、SYSPCI ユーティリティはファイルの構造（FDT）が正しいかどうかのみを確認します。 新規ファイルの場合も既存ファイルの場合も、次のアクションが実行されます。 ■ 必要な各 Adabas ファイルのデータベース ID が Natural のグローバルコンフィグレーションファイルに入力されます。 ■ 新規または既存の各 Adabas ファイルのデータベース ID とファイル番号が製品のデフォルトパラメータファイルに入力されます。
Initialize product	このオプションは、初期化プログラムを持つ製品にのみ使用できます。 マークされている場合（デフォルト）、選択した製品の初期化プログラムがロード、実行されます。製品を有効にする場合は、このオプションをマークする必要があります。 **注意:** このオプションを使用する場合は、データベース ID を指定したデータベースがオンラインである必要があります。
データベース ID	Adabas ファイルのデータベース ID。 ［**Database ID**］の隣にある［**Select**］フィールドをマークするか PF5 キーを押すと、マシン上にあるすべてのデータベースのリストが表示されます。リストには、

オプション	説明
	データベースが現在オンラインかオフラインかも表示されます。その後、リスト内のデータベースにマークを付けて選択できます。
File number	選択したデータベース内のファイル数。既存のファイルの数または新しいファイルの数です。 ［**File number**］の隣にある［**Select**］フィールドをマークするか PF6 キーを押すと、指定したデータベース ID に対して見つかったすべてのファイルのリストが表示されます。その後、リスト内のファイルにマークを付けて選択できます。 Adabas システムファイルはリストにも表示されるため、どのファイル番号がすでに割り当てられているかを確認できます。ただし、Adabas システムファイルをマークしないでください。そうでない場合、エラーが発生します。 新しいファイルの番号を指定するときは、［**Create new Adabas file**］オプションがマークされていることを確認してください。
File name	Adabas ファイルの名前。 既存のファイルを選択した場合、対応するファイル名が自動的に表示されます。この名前は変更できません。 まだ存在しない新しいファイル番号を指定した場合は、ファイル名を入力できます（オプション）。ファイル名を入力しない場合は、製品固有のデフォルト名が使用されます。

Predict など一部の製品では、追加オプションを指定する必要があります。詳細については、その製品のインストールドキュメントを参照してください。

オプションの情報を入力したら、ENTER キーまたは PF4 キーを押して入力を検証できます。

5 ［**Start selected action(s)**］フィールドで "Y" と入力して ENTER キーを押して、選択した製品の設定を開始します。

選択した製品に対して選択したアクションが実行されると、次のようなメッセージが表示されます。

```
Function completed successfully.

The following actions have been performed by the SYSPCI utility:

- Loaded Adabas file with DBID 10 FNR 55 for product NCP
- Updated global configuration file for DBID 10
- Updated Natural parameter file NATPARM and set LFILE 190 to DBID 10 FNR 55
```

6 ENTER キーを押して次に進みます。

SYSPCI ユーティリティの最初の画面が再度表示されるため、製品を追加設定できます。ただし、Natural Security の設定は例外です。この場合は、初期化プログラムが実行された後に Natural が終了します。

ダイレクトコマンドデータを使用したSYSPCIユーティリティの呼び出し

SYSPCI ユーティリティは、キーワードと対応する値で構成されるダイレクトコマンドを使用して呼び出すことができます。そのため、バッチモードでSYSPCIユーティリティを使用することもできます。

例外：Natural Security の Adabas ファイルの初期化が（INPL ユーティリティで初期化プログラムをロードして実行することによって）完了すると、Natural セッションが INPL ユーティリティによって終了します。したがって、この手順の後で追加コマンドを実行することはできません。

SYSPCI コマンドでは、次のキーワードを使用できます（次の例も参照）。

キーワード	意味
PRODUCT*	処理する製品。有効な値： NDV：Natural 開発サーバーの場合 NCP：Natural コマンドプロセッサの場合 PRD：Predict の場合 CST：Construct の場合（NBS の代替） NBS：Natural Business Services の場合 SAT：System Automation Tools の場合 NOM：Entire Output Management の場合 NOP：Entire Operations の場合 NSL：Natural Security Log の場合 NSC：Natural Security の場合 **重要:** 上記の順序で製品を設定することをお勧めします。
DBID*	Adabas ファイルのデータベース ID。 **注意:** FUNCTION ADU または ALU を使用する場合はこれを空白のままにできます。この場合、SYSPCI ユーティリティは SYSPCI ライブラリ内の INST-<*productcode*> テキストメンバで見つかった DBID 値を使用します。
DBID2**	2 番目の Adabas ファイルのデータベース ID（PRODUCT が PRD、NBS、NOM、または NOP の場合）。
DBID3**	3 番目の Adabas ファイルのデータベース ID（PRODUCT が NBS の場合）。
FNR*	Adabas ファイルのファイル番号。 **注意:** 1. FUNCTION ADU または ALU を使用する場合はこれを空白のままにできます。この場合、SYSPCI ユーティリティは SYSPCI ライブラリ内の INST-<*productcode*> テキストメンバで見つかった FNR 値を使用します。

キーワード	意味
	2. FUNCTION ADA または ALL を使用する場合は、-1 を入力できます。この場合、SYSPCI ユーティリティは次の空きファイル番号を使用します。
FNR2 **	2 番目の Adabas ファイルのファイル番号（PRODUCT が PRD、NBS、NOM、または NOP の場合）。
FNR3 **	3 番目の Adabas ファイルのファイル番号（PRODUCT が NBS の場合）。
FUNCTION または FCT	実行される機能。有効な値： ADA：新しい Adabas ファイルを作成します。 ADU：既存の Adabas ファイルを使用します。これには、有効な DBID と FNR（上記を参照）を入力するか、値を入力する必要があります。 INT：初期化プログラムをロードして実行します。 ALL：両方（ADA および INT）。デフォルト値。 ALU：両方（ADU および INT）。
FILE-NAME	Adabas ファイルの名前（FUNCTION が ADA または ALL の場合）。有効な値：空白を含まない 16 文字。
FILE-NAME-2 または FILE-N2	2 番目の Adabas ファイルの名前（PRODUCT が PRD、NBS、NOM、または NOP の場合）。有効な値：空白を含まない 16 文字。
FILE-NAME-3 または FILE-N3	3 番目の Adabas ファイルの名前（PRODUCT が NBS の場合）。有効な値：空白を含まない 16 文字。
SUBFUNCTION	実行される追加機能。有効な値は製品によって異なります。 PRODUCT が PRD である場合： ■ PRC：FDIC データを変換します。 ■ PRP：FDIC の記述をロードします。 ■ PRD：サンプルデータをロードします。 ■ PRA：両方（PRP および PRD）。 ■ PR1：FDIC データを変換して FDIC の記述をロードします。 ■ PR2：FDIC データを変換してサンプルデータをロードします。 ■ PR3：FDIC データを変換して FDIC の記述とサンプルデータをロードします。 PRODUCT が NBS または CST の場合： ■ LDC：Construct データをロードします。 ■ LDP：Predict データをロードします。 ■ LDA：両方（LDC および LDP）。 PRODUCT が NOM である場合： ■ LDM：NOM データをロードします。

キーワード	意味
END、STOP、EXIT、QUIT または .	SYSPCI ユーティリティを終了します。キーワードは単一のコマンドとして入力する必要があります。
FIN	SYSPCI ユーティリティと Natural セッションを終了します。キーワードは単一のコマンドとして入力する必要があります。

注意:

1. アスタリスク（*）が付いているキーワードは必須です。
2. 対応する製品には、アスタリスクが 2 つ付いているキーワード（**）が必須です。
3. その他のキーワードはすべてオプションです。

例

- バッチモード
  CMSYNIN プロファイルパラメータによって定義されるバッチ入力ファイル内のコマンド：

```
SYSPCI
FIN
```

  CMOBJIN プロファイルパラメータによって定義されるバッチ入力ファイル内のデータ：

```
FUNCTION ALL PRODUCT PRD DBID 77 FNR 2002 DBID2 12 FNR2 2003
FUNCTION ALL PRODUCT NSC DBID 77 FNR 1600
END
```

  『*オペレーション*』ドキュメントの「*バッチモードでの Natural*」も参照してください。
- 対話式モード **- Natural** コマンド行

```
SYSPCI FUNCTION ALL PRODUCT NSL DBID 77 FNR 1601
```

- 対話式モード **- Natural** スタック

```
natural stack='(SYSPCI FUNC ALL PROD NCP DBID 77 FNR 1501: PROD NSL DBID 77 FNR ↩
1601; FIN)'
```

XII

- 68 Natural Profiler 449
- 69 Profiling Natural Applications 451
- 70 Code Coverage of Natural Applications 463
- 71 Basic Concepts of the Profiler Utility 475
- 72 Using the Profiler Utility 485
- 73 Natural Profiler MashApp 539

68 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.
Using the Profiler Utility	Evaluating the event data from the Profiler resource files and code coverage.
Natural Profiler MashApp	Evaluating Profiler data on an interactive MashZone dashboard.

注意: The features of the NaturalONE Profiler and NaturalONE code coverage are described in the relevant sections of the NaturalONE documentation. The use of the Natural Profiler for UNIX and Windows is described with the `PROFILER` profile parameter in the Natural *Parameter Reference* documentation. The use of Natural code coverage for UNIX and Windows is described with the `COVERAGE` profile parameter in the Natural *Parameter Reference* documentation.

69 Profiling Natural Applications

- Introducing Profiling 452
- Platform-Specific Profiling 452
- Profiling Tools 453
- Natural Profiler Evaluations 455

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.

注意: 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.

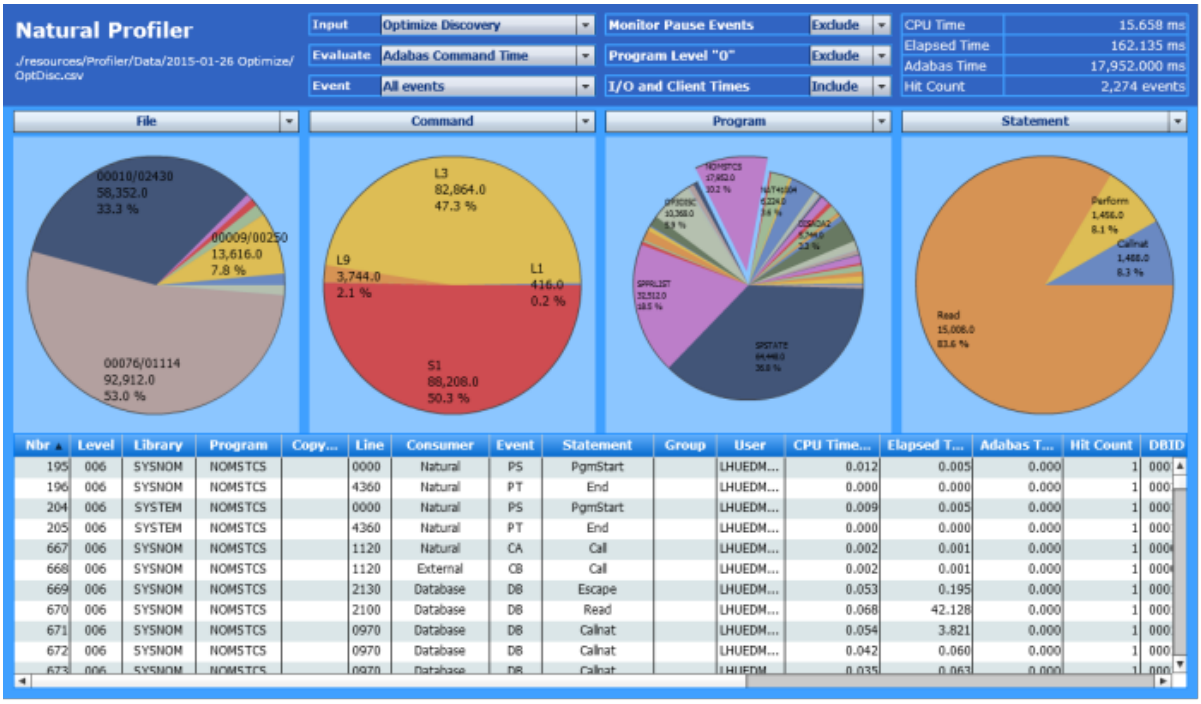
Evaluation Page

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.

70 Code Coverage of Natural Applications

- Introducing Code Coverage 464
- Platform-Specific Code Coverage 465
- Code Coverage Tools 465
- Natural Code Coverage Evaluations 468

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.

注意: 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.

注意: 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.

注意: 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).

注意: 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).

注意: 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
```

71 Basic Concepts of the Profiler Utility

- Data Consolidation, Code Coverage and Data Processing 476
- Sampling 479
- Profiling Long-Running Applications 481
- Related Topics 484

The Profiler utility reads and processes Profiler resource files created by the Natural Profiler for UNIX and Windows and Natural code coverage for UNIX and Windows. 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 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**.

Additionally, the Profiler utility provides functions to pause and to restart the Profiler data collection.

The Profiler utility runs in batch mode only.

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 off the `EVENTTRACE` subparameter of the `PROFILER` parameter. See *PROFILER - Profile a Natural Session* in the *Parameter*

Reference documentation. 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.

注意:

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

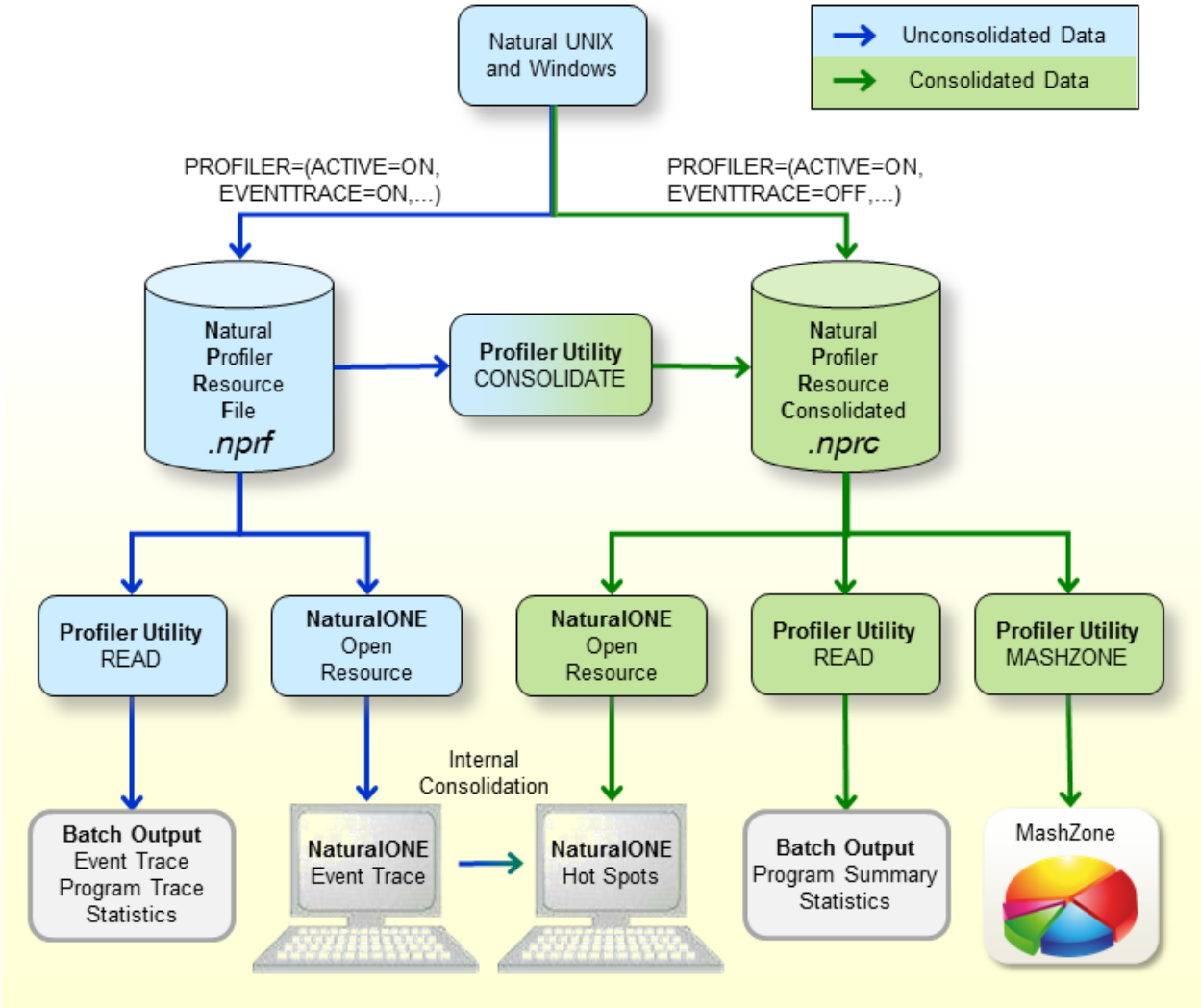
Natural code coverage is used to monitor executed and not-executed statements of a Natural application. It is started by switching on the `ACTIVE` subparameter of the `COVERAGE` profile parameter described in the *Parameter Reference* documentation.

For code coverage, Natural code coverage automatically uses an event filter so that only the program information (`PI`) and Natural statement (`NS`) events are collected. The data is automatically consolidated before it is 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.

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 application on UNIX or Windows is profiled by switching on the `ACTIVE` subparameter of the `PROFILER` profile parameter, 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 `EVENTTRACE` subparameter of the `PROFILER` parameter. For `EVENTTRACE=ON`, the data is written to an NPRF resource file, for `EVENTTRACE=OFF`, it is written to an NPRC resource file.
- 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.

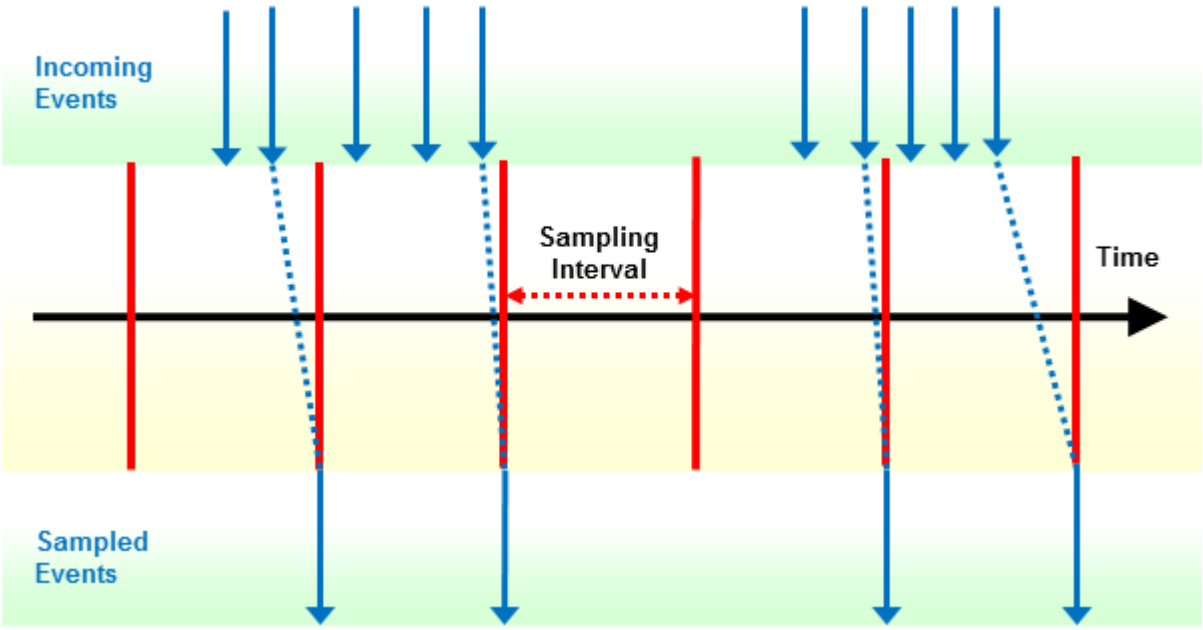
- The Profiler utility `READ` function reads the event data from the NPRF resource file. It provides an event trace, a program trace and the Profiler statistics. The resulting data can be exported to a file in text or CSV (comma-separated values) format.
- If the NPRF resource file is opened from NaturalONE, 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:

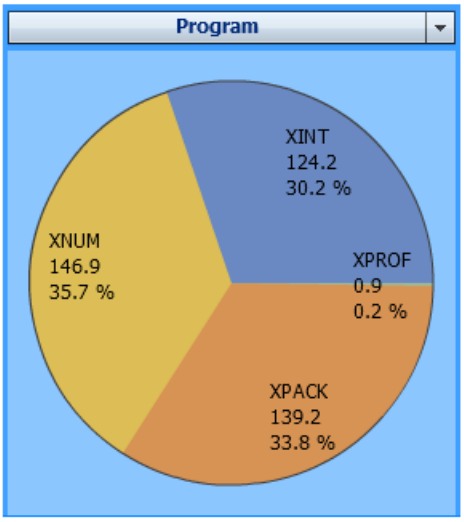
1. Without sampling.

2. With sampling whereby a sampling interval of 100 microseconds is used.

For sampling, the following subparameters of the `PROFILER` profile parameter are used:

```
PROFILER=(ACTIVE=ON,SAMPLING=ON,INTERVAL=100)
```

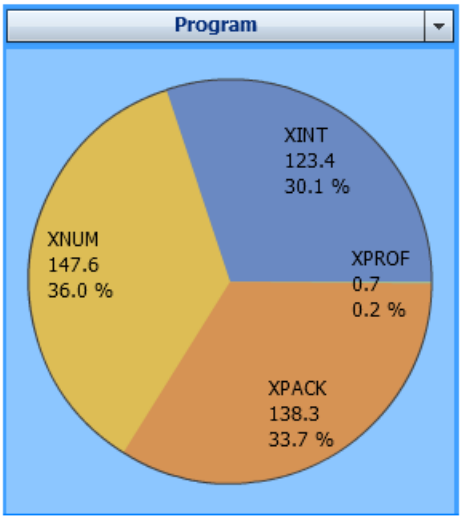
The Natural Profiler MashZone pie charts below show for each program the name of the program, the CPU time spent (in units of milliseconds) and the CPU time percentage with respect to the total CPU time. The left chart reflects the run without sampling and the right chart the run with sampling. Although 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.



Without Sampling

411.4 ms total CPU time

240,086 events



Sampling 100 µs

410.0 ms total CPU time

4,100 events

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 a Natural session executes multiple Natural applications, pause the Profiler for applications which are not of interest and restart it for applications of interest.
- Eventually, use the application programming interface (API) to start and pause profiling at specific points in the application.

Example

A Natural batch session executes three Natural applications. From these three applications, only the second one is of interest for a Profiler analysis.

Pause profiling before the first application executes, start profiling before the second application executes, and pause profiling again before the third application executes as in the example below:

```
PRFPAUSE
APP-01
PRFSTART
APP-02
PRFPAUSE
APP-03
FIN
/*
```

This way, profiling only affects the second application and has no impact on the performance of the other applications. Note that the programs `PRFPAUSE` and `PRFSTART` have to be copied into the user library.

Set Filters

- 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 setting of the `PROFILER` profile parameter is sufficient:

  ```
  PROFILER=(ACTIVE=ON,EVENT=(S,P),...)
  ```

  With this setting, only the program and session events needed for the program hot spots are monitored.

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.

$$\text{Sampling interval} \geq \frac{2{,}400{,}000{,}000}{500{,}000} = 4{,}800$$

Set the `PROFILER` profile parameter as follows:

```
PROFILER=(ACTIVE=ON,SAMPLING=ON,INTERVAL=4800)
```

See *PROFILER - Profile a Natural Session* in the *Parameter Reference* documentation.

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 by switching off the `EVENTTRACE` subparameter of the `PROFILER` profile parameter. See *PROFILER - Profile a Natural Session* in the *Parameter Reference* documentation.

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.

注意: Natural code coverage data written to an NCVF resource file is automatically consolidated by Natural code coverage.

Related Topics

- The Natural Profiler for UNIX and Windows is activated with the `PROFILER` profile parameter described in the *Parameter Reference* 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.

72 Using the Profiler Utility

- Quick Start for Profiling 486
- Quick Start for Code Coverage 489
- Prerequisites 491
- Invoking and Terminating the Profiler Utility 492
- Syntax and Keywords 492
- Events and Data Collected 496
- Starting and Pausing Data Collection 500
- Consolidating Event Data 504
- Evaluating Event Data 506
- Exporting Event Data for MashZone 521
- Maintaining Profiler Resource Files 523
- Including Profiler Input from Natural Text Objects 525
- Event Trace 526
- Tracing Natural Code Coverage 528
- Internal Trace 529
- Profiler Statistics 530

The Natural Profiler is used to monitor the internal process flow of a Natural application and to analyze the performance and the code coverage of the application. Profiling of Natural applications is activated by switching on the `ACTIVE` subparameter of the `PROFILER` profile parameter. The Natural Profiler writes the collected Profiler event data to a Profiler resource file. See *PROFILER – Profile a Natural Session* in the *Parameter Reference* documentation.

Code coverage of Natural applications is activated by switching on the `ACTIVE` subparameter of the `COVERAGE` profile parameter (see the *Parameter Reference* documentation). Natural code coverage writes the collected coverage data to a code coverage resource file.

The Profiler utility runs in batch mode only. It provides functions to control the Profiler and code coverage data collection and to process the resulting data.

1. With the Profiler data collection functions, the data collection can be paused and restarted (see also ***Starting and Pausing Data Collection***).
2. The **data processing** functions read and process the event data from the Profiler resource file. Unconsolidated event data can be consolidated.

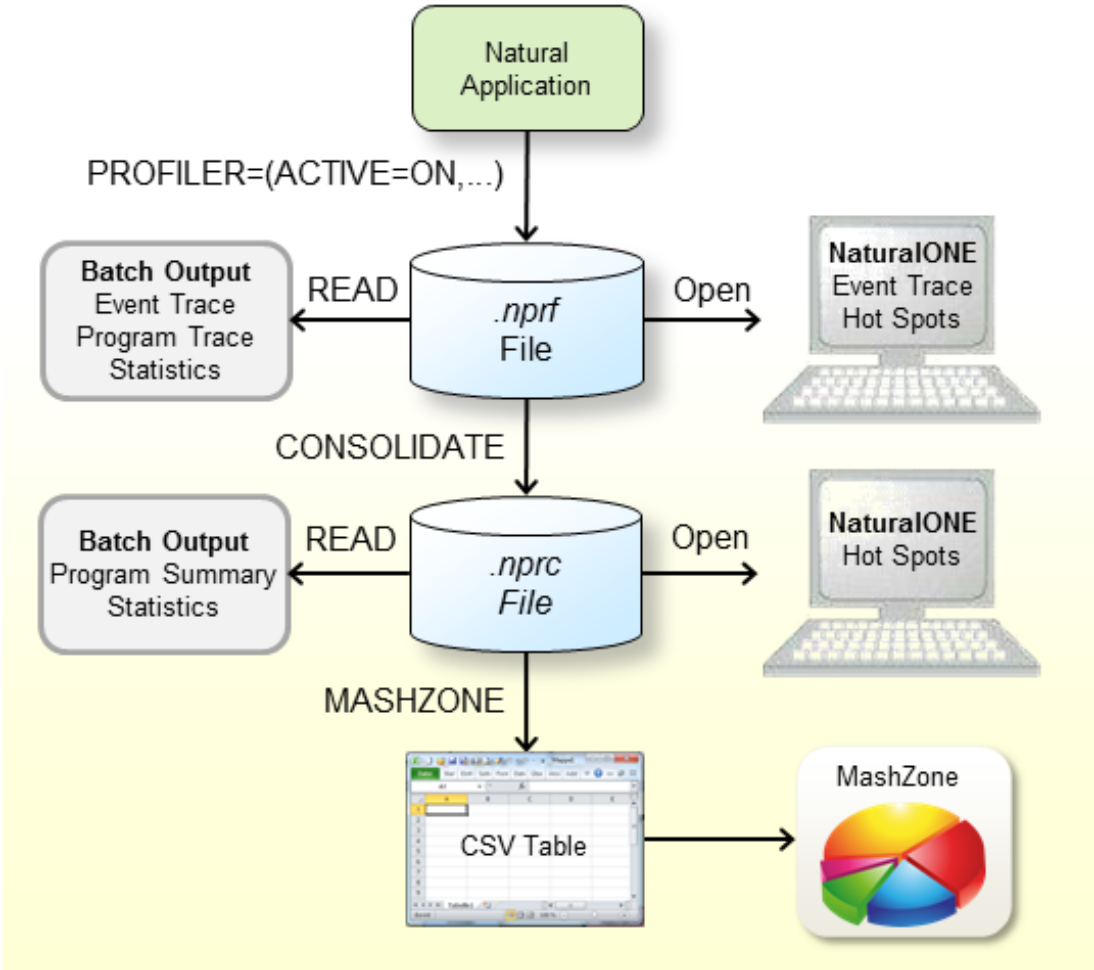
   You can output statistics, a program summary, a program trace, an event trace with the most important data, and reports on program and statement coverage. You can export the resulting data in text or CSV (comma-separated values) format.

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 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. Activate the profiling of the Natural session by switching on the `ACTIVE` subparameter of the `PROFILER` profile parameter in the NATPARM parameter file or dynamically when invoking Natural. Example for dynamic parameter specification:

   ```
   PROFILER=(ACTIVE=ON,RESNAME=ResName,RESLIB=RESLIB)
   ```

   In the example above, the Profiler event data is written to a resource file with the name `ResNam.nprf` in the library `RESLIB`. See *PROFILER – Profile a Natural Session* in the *Parameter Reference* documentation.
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. Use as Work File 7 a CSV (comma-separated values) file in the Natural Profiler data directory in the MashZone environment. Example:

```
FUNCTION=MASHZONE        /* Write MashZone format to Work File 7
  RESOURCE-LIB=RESLIB    /* Resource library
```

See also *Exporting Event Data for MashZone*.

9. 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.

注意:

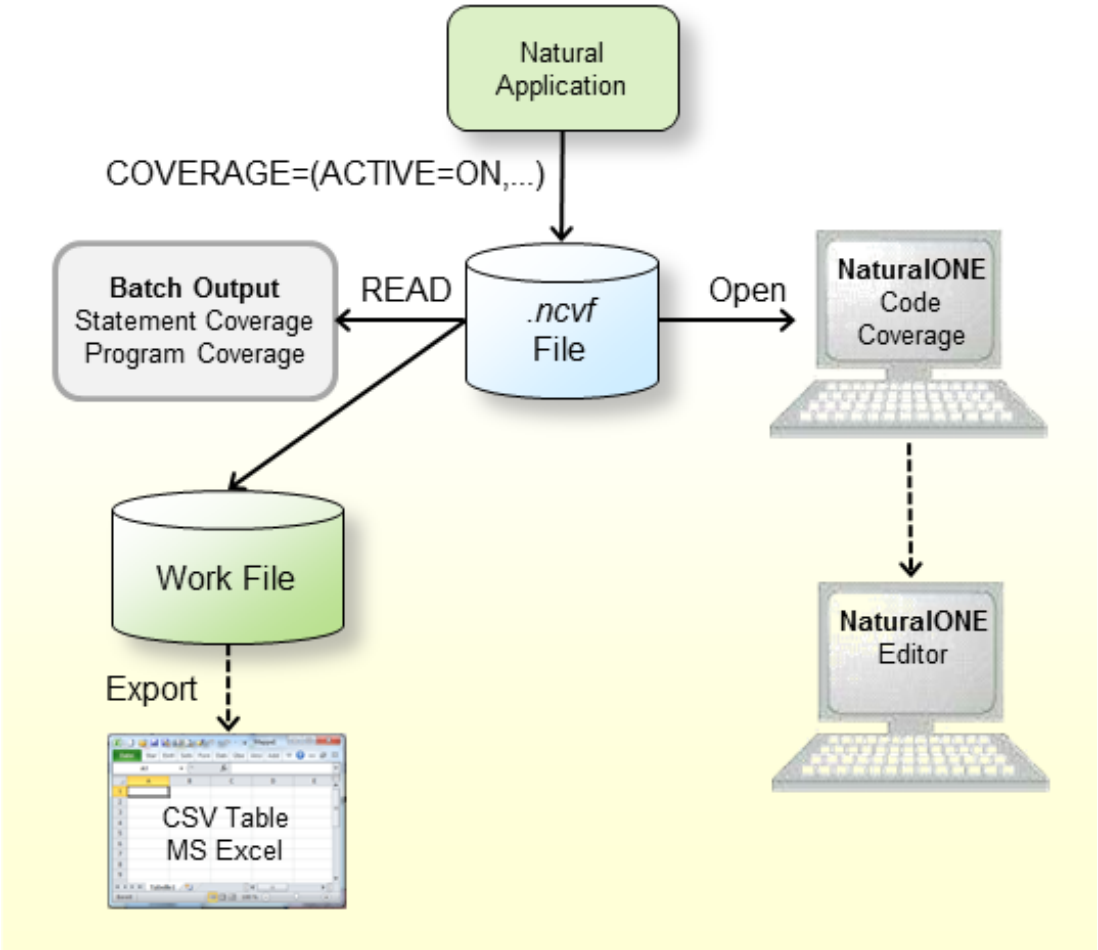
1. If the resource name is not explicitly specified in the `READ`, `CONSOLIDATE` or `MASHZONE` function of the Profiler utility, the last created NPRF or NPRC resource in the library is used.

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:



1. Check that the prerequisites are met.

2. Activate code coverage of the Natural session by switching on the `ACTIVE` subparameter of the `COVERAGE` profile parameter (see the *Parameter Reference* documentation) in the NATPARM parameter file or dynamically when invoking Natural. Example for dynamic parameter specification:

   ```
   COVERAGE=(ACTIVE=ON,RESNAME=ResName,RESLIB=RESLIB)
   ```

   In the example above, the coverage data is written to a resource file with the name `ResNam.ncvf` in the library `RESLIB`.
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.

   Example:

   ```
   FUNCTION=READ            /* Read Profiler data
     RESOURCE-LIB=RESLIB    /* Resource library
     RESOURCE-TYPE=NCVF     /* Use resource type
     EVENT=ON               /* Print statement coverage
     PROGRAM=ON             /* Print program coverage
     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.

注意:

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 prerequisite must be met before you can use the Profiler utility:

- Natural Parameter Settings
- Activate Data Processing

Natural Parameter Settings

The Profiler utility data processing functions (`CONSOLIDATE`, `READ`, `MASHZONE` and `LIST`) cannot be executed if profiling is active. Deactivate the profiling infrastructure with the following (default) parameter setting:

```
PROFILER=(ACTIVE=OFF)
```

For details regarding the `PROFILER` parameter, see *PROFILER – Profile a Natural Session* in the *Parameter Reference* documentation.

Activate Data Processing

If NaturalONE is installed at your site, you can activate the Profiler utility data processing functions (`CONSOLIDATE`, `READ`, `MASHZONE` and `LIST`) 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.

手順 72.1. To invoke the Profiler utility

- Enter the following system command into the primary command input data set `CMSYNIN`:

```
PROFILER
```

注意: After the `PROFILER` system command, the Profiler expects one or more lines with Profiler keyword entries.

手順 72.2. To terminate the Profiler utility

- Enter the following Profiler keyword into the primary command input data set `CMSYNIN`:

```
END-PROFILER
```

または:

```
END
```

または:

```
.
```

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]
...
```

注意:

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 `START` function, and then, after the execution of a user program, it is executed with the `PAUSE` function.

Example

The following Natural batch example runs the Profiler utility READ function:

```
natural BATCHMODE CMSYNIN=cmd.txt CMOBJIN=data.txt CMPRINT=out.txt CMWRK07=wrk07.txt
```

The content of the batch input file cmd.txt which contains the PROFILER command is shown below:

```
PROFILER
FIN
```

The content of the input file data.txt which contains the input for the Profiler utility is shown below:

```
*******************************************************************
* Read Profiler NPRF Resource
*******************************************************************
TRACE=3                  /* Set Profiler trace level
FUNCTION=READ            /* Read Profiler resource
  RESOURCE-NAME='Demo01' /* Resource name
  RESOURCE-LIB=PRFDATA   /* Resource library
  RESOURCE-TYPE=NPRF     /* Resource type
  EVENT=ON               /* List events
  STATISTICS=ON          /* List properties and statistics
  PROGRAM=ON             /* Program trace
  PRINT=ON               /* Write to standard output
  EXPORT=ON              /* Write to work file 7
  FORMAT=TEXT            /* Use text format
END-PROFILER             /* End profiler input
```

After execution, out.txt contains the Profiler utility output and the internal trace (TRACE=3). wrk07.txt contains the Profiler utility output in text format (FORMAT=TEXT).

The following Natural example demonstrates how the Profiler utility PAUSE function is to be executed with the Natural STACK profile parameter:

```
natural PROFILER=(ACTIVE=ON,RESNAME=ResName,RESLIB=RESLIB)
STACK=(
PROFILER FUNCTION:PAUSE:END-PROFILER;
LOGON PRFDEMO
)
```

After execution, profiling of the Natural session is activated but the data collection is paused. The data collection can be started later with the Profiler utility START function.

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***.
	`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***.
`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***. 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. See also ***Including Profiler Input from Natural Text Objects***.
`INCLUDE-LIB`	*`library-name`*	The name of the Natural library that contains the text object specified with the `INCLUDE` keyword. If the Natural system variable `*LIBRARY-ID` is specified, the name of the current library is used. The library name is used for all following `INCLUDE` keywords. Default: If `INCLUDE-LIB` is not specified before an `INCLUDE` keyword, the Natural system library `SYSPRFLR` is used by default.

Keyword	Value	Description
		See also *Including Profiler Input from Natural Text Objects*.
END-PROFILER or END or .		End of Profiler input. The keyword END-PROFILER, END or a period (.) indicates the end of the Profiler input.

Events and Data Collected

This section describes the events and data processed by the Profiler utility.

- 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
Session **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.
Session **T**ermination	ST	S	When a Natural batch session is terminated. The Profiler always monitors this event.
Program **L**oad	PL	P	When a program (Natural object) is loaded or when it is already located in the buffer pool.
Program **S**tart	PS	P	When a program (Natural object) is started.
Program **T**ermination	PT	P	When a program (Natural object) is terminated.
Program **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).

Event	Event Type	Event Group	When the Event Occurs
Program **I**nformation	PI	P	When a program (Natural object) is accessed for the first time. This event is only triggered at Natural code coverage.
Before **D**atabase Call	DB	D	Before a database call is executed.
After **D**atabase Call	DA	D	After a database call has been executed.
Before Terminal **I**/O	IB	I	Before a terminal input/output is executed.
After Terminal **I**/O	IA	I	After a terminal input/output has been executed.
Before External Program **C**all	CB	C	Before an external program call (CALL statement) is executed.
After 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.
Natural **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.
Inbound **R**PC Message	RI	R	When the Natural RPC server layer receives the client request.
Start of **R**PC Request Execution	RS	R	When the Natural RPC server layer calls the Natural server program.
Outbound **R**PC Message	RO	R	When the Natural RPC server returns the result to the client.
RPC **W**ait for Client	RW	R	When the Natural RPC server waits for the next message from the client.
User-Defined **E**vent	U	U	When a user-defined event was generated.
Monitor **P**ause	MP	M	When the data collection is paused. 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 Natural Profiler:

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)

注意:

1. The Natural Profiler for UNIX and Windows does not yet collect RPC-related events.
2. Natural code coverage only collects `NS` and `PI` events.
3. Natural code coverage does not collect time values.
4. 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 Natural termination message code NAT99*nn* Name of back-end program Monitor CPU time in units of microseconds
Program Load	Name of program to be loaded Name of load library Invocation type
Program Resume	None
Program Start/Termination	Program type Database ID of program library File number of program library
Program Information	Program type Number of statements in the program or copycode First statement item `INCLUDE` line number Parent copycode ID
Database Call	Database type Command code Command ID Database ID File number Response code (event type `DA`) Error subcode (event type `DA`) Adabas command time (event type `DA`)
Terminal I/O	Number of bytes sent Number of bytes read Total session storage allocated Compressed session storage length
External Program Call	Name of program called Calling mode such as dynamic or static mode Program link location Parameter type such as reference or value Response code (event type `CA`)
Runtime Error	Natural system error message code External abend code Name of error handling program
Natural Statement	Profiling: None Natural code coverage: Statement item identifier (GP offset)
Start of RPC Request Execution	Environment (`C` = client, `S` = server) Subprogram name Adabas user ID (ETID) Conversation status

Event	Data Elements	
	Logon indicator (Y = logon performed) Impersonation indicator of RPC request (Y = impersonation performed)	
Outbound/Inbound RPC Message / RPC Wait for Client	Environment (C = client, S = server) Transport protocol RPC function Type of client user ID Length of message RPC return code External conversation ID Client user ID Server node (event types RO and RW) Server name (event types RO and RW)	
User-Defined Event	Subtype of the user-defined event Up to 249 bytes of user-defined information	
Monitor Pause	Type of monitor pause 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.

Starting and Pausing Data Collection

When a Natural session is profiled, all event data of the session is collected by default.

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. Valid values for *count-number*: 0 to 2147483647 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.

手順 72.3. 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.

  注意: PRFFCT expects as input the value START, PAUSE or STATE to perform the corresponding function.

手順 72.4. To start the data collection

- Execute the following program:

```
PRFSTART
```

または:

```
PRFFCT
START
```

手順 72.5. To pause the data collection

- Execute the following program:

```
PRFPAUSE
```

または:

```
PRFFCT
PAUSE
```

手順 72.6. To retrieve the current state of the data collection

- Execute the following program:

```
PRFSTATE
```

または:

```
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*.

手順 72.7. To use the API

- Copy the subprogram USR8210N to the library SYSTEM, to the appropriate steplib library, or to the required library.

  注意: 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.

手順 72.8. To start the data collection

- Use the interface with the CALLNAT statement:

```
CALLNAT 'USR8210N' 'START' P-RETURN P-MESSAGE /* Start Profiler
```

手順 72.9. To pause the data collection

- Use the interface with the CALLNAT statement:

```
CALLNAT 'USR8210N' 'PAUSE' P-RETURN P-MESSAGE /* Pause Profiler
```

手順 72.10. 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.

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. 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. The file extension `.nprf` is added automatically. 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. Default: The name of the current library. 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. 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.
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. See *Event Trace*.
	ON	Write the Profiler event trace.
	OFF	Do not write the Profiler event trace.
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 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

Resource File Type	Evaluation	Description
NCVF	Statement coverage	List of statements covered in in the source lines 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 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. If no file extension is specified, the extension specified with the keyword RESOURCE-TYPE is added automatically. 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. 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. See also ***Event Trace***, ***Consolidation Trace*** and ***Statement Coverage***.
	ON	NPRF: Write the Natural Profiler event trace. NPRC: Write the Natural Profiler consolidation trace. NCVF: Write the statement coverage result.
	OFF	Do not evaluate events.
PROGRAM		Specifies whether the Natural Profiler evaluates programs. See also ***Program Trace***, ***Program Summary*** and ***Program Coverage***.
	ON	NPRF: Write the Natural Profiler program trace. NPRC: Write the Natural Profiler program summary. NCVF: Write the program coverage table.
	OFF	Do not evaluate programs.
STATISTICS		Specifies whether the Natural Profiler writes statistics. See also ***Profiler Statistics***. **注意:** 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.

Keyword for READ	Value	Description
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.
- 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 `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 PL PRF      000                   0000                                       1        0.751        0.752
        2 PR PRF      000                   0000                                       1        0.549        0.534
        3 SI PRF      000                   0000                                       1       30.916       25.599
        4 ST PRF      000                   0000                                       1        0.000        0.000
        5 NS PRF      002 PRFDEMO  XINT     0140                   Reset               1        0.002        0.003
        6 NS PRF      002 PRFDEMO  XINT     0150                   Assign/com          1        0.002        0.001
        7 NS PRF      002 PRFDEMO  XINT     0160                   Assign/com          1        0.000        0.000
        8 NS PRF      002 PRFDEMO  XINT     0170                   For                31        0.027        0.026
...
```

Explanations:

- The **Count** column shows the number of the consolidated record.
- The consolidated record `8` shows that the `FOR` statement in the line `0170` of the Natural object `XINT` executed `31` times spending a total elapsed time of `0.027` milliseconds (`ms`) and a total CPU time of `0.026 ms`.

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. The copycode ID of the object or copycode that includes the copycode.
INCLUDE Object	For copycode only. The name of the object or copycode that includes the copycode.
INCLUDE Line	For copycode only. 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      78.5%       11         3       14
COVDEMO   TESTCOVP P      57.1%        4         3        7
COVDEMO   -------- --     71.4%       15         6       21
Totals    -------- --     71.4%       15         6       21
```

Explanations:

- The application accesses two objects, the TESTCOVN subprogram (N) and the TESTCOVP program (P).
- In TESTCOVN, there are 14 executable statements from which 11 were covered (executed) and 3 missed (not executed), giving a total coverage of 78.5%.
- The summarized values of the two objects accessed in the library COVDEMO show coverage of 71.4%.
- Total coverage is also 71.4% 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. 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):

```
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
+                        0130   VALUE 0
+                        0140     PRINT 'Test function 0'
+                        0150   VALUE 1
C 100%                   0160     PRINT 'Test function 1'
+                        0170   VALUE 2
C 100%                   0180     PRINT 'Test function 2'
+                        0190   VALUE 3
C 100%                   0200     PRINT 'Test function 3'
+                        0210   VALUE 4
C 100%                   0220     PRINT 'Test function 4'
+                        0230   VALUE 5
C 100%                   0240     PRINT 'Test function 5'
```

```
+                         0250   VALUE 6
C 100%                    0260     PRINT 'Test function 6'
+                         0270   VALUE 7
C 100%                    0280     PRINT 'Test function 7'
+                         0290   VALUE 8
C 100%                    0300     PRINT 'Test function 8'
+                         0310   VALUE 9
+                         0320     PRINT 'New test function 9'
+                         0330   NONE VALUE
+                         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`).
- No test cases cover the functions `Test function 0` and `New test function 9`. The `NONE VALUE` case is also not covered.
- All other test cases are covered (denoted with `C` and 100% coverage).

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. 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	1	0	1	x1	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'		0	0	0		+
1	QFTEST	TESTCOVC	10	* Test Coverage		0	0	0		*
1	QFTEST	TESTCOVC	20	* Copycode TESTCOVC		0	0	0		*
1	QFTEST	TESTCOVC	30	IF &1& > 0	100	1	0	1	x1	C
1	QFTEST	TESTCOVC	40	IF &1& = 1		0	0	0		+
1	QFTEST	TESTCOVC	50	PRINT 'Unsupported function' &2&		0	0	0		+
1	QFTEST	TESTCOVC	60	ELSE		0	0	0		+
1	QFTEST	TESTCOVC	70	PRINT 'Return code:' &1&		0	0	0		+
1	QFTEST	TESTCOVC	80	END-IF		0	0	0		+
1	QFTEST	TESTCOVC	90	END-IF		0	0	0		+
0			110	END-FOR		0	0	0		+
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 of the copycode contain statements which were not executed in the test run.
- All other lines (in green) containing executable statements are covered.

Statistics

If `STATISTICS=ON` is specified, the Profiler statistics are listed.

注意: Statistics are not provided for an NCVF file on UNIX or Windows.

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:

```
************************************************************************
* 17:35:59          ***** NATURAL PROFILER UTILITY *****         2016-01-11
* User SAGTEST1                - Statistics -                        RESDATA
...
*
* 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
...
*
* 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 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*.

注意: The Natural Profiler for UNIX and Windows does not collect statistical data. The statistics provided are values determined by the Profiler utility.

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. Use as

Work File 7 a CSV (comma-separated values) file in the Natural Profiler data directory in the MashZone environment which 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. The extension .nprc is added automatically. 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. 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 accessed by 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 Profiler resource files.

This section covers the following topic:

- Listing Profiler Resource Files

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. Default: The name of the current library
RESOURCE-TYPE		Specifies the type of resource files to be listed: NPRF, NPRC or NCVF. 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
```

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*).

手順 72.11. 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 content of the Natural text member `XCONS` from the library `SYSPRFLR` to the Profiler input data. The object consolidates Profiler event data. Additionally, it terminates the Profiler utility so that no further Profiler input is expected after the `INCLUDE` keyword.

```
INCLUDE=XCONS
```

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 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.

注意: The event trace can also be listed in NaturalONE.

手順 72.12. To enable the event trace

- 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 Unit: *hour*:*minute*:*second*.*microseconds*
CPU-Time	Session CPU time 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). This data is only displayed for the Profiler utility `READ` function.
Elapsed (ms)	Elapsed time spent processing the event. Unit: milliseconds This data is only displayed for the Profiler utility `CONSOLIDATE` function.
CPU-Delta	CPU time spent processing the event. 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 the coverage resource is read with the Profiler utility `READ` function, the coverage data can be traced with the internal data trace.

手順 72.13. To enable tracing for code coverage

- 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 trace:

Property	Description
Count	The event count.
Ev	The event type. See *Events and Data Collected*.
Library	The name of the Natural library that contains the program/object.
Program/Object	The name of the Natural program/object.
Ty	The object type such as `P` for program.
CC-Lib	The name of the Natural library that contains the copycode (if copycode is active).
CC-Name	The name of the copycode.
Line	The source line number.
CC-ID	The copycode ID. It uniquely identifies the copycode instance in the GP. The program gets the copycode ID `0`.

Property	Description
Par-CC	For copycode only. The parent copycode ID which is the copycode ID of the object/copycode that includes the current copycode.
FirstS	The ID of the first statement of the object or copycode.
Stmts	The total number of executable statements in the object whereby all `INCLUDE` statements are resolved.
Item	The item ID of the statement. It uniquely identifies the statement in the resource file.
Cover	The coverage flag (`0` or `1`) of the statement. When the GP is read, all flags are initialized with `0`. Whenever a statement is executed, the flag is set to `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 data processing functions. The data is written to standard output.

手順 72.14. To activate the internal trace for the Profiler data processing functions

- Enter the following Profiler keyword:

```
TRACE=n
```

where *n* is the trace level (see ***Trace Levels***).

注意:

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.

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.

Trace Level	Name	Description
0	No trace	Profiler internal trace is deactivated.
1	Error	Log error messages.
2	Warning	Log warnings.
3	Statistics	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):

```
* Set Profiler internal trace
TRACE=4                      /* Trace level
```

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 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 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

注意: 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`).

Property	Unit	Description
Profiler version		The internal version of the Profiler. 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.

Property	Unit	Description
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.

Property	Unit	Description
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.

Property	Unit	Description
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.
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.

Property	Description
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.
Consolidation factor		The average number of events combined into one consolidation record. The higher the consolidation factor, the better the consolidation. `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.

注意: 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.

73 Natural Profiler MashApp

- Preparing to Use the MashApps 540
- Preparing the Profiler Data 544
- Opening the MashApps 545
- Evaluation Page 546
- Compare Page 555
- Properties Page 557
- Use Cases 559

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.

手順 73.1. 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.

手順 73.2. 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. *date* is the MashApp generation date.
`resources\Profiler`	Parent directory of Profiler resources. Contains the user-modified `Overview.csv` resource file. See also ***Editing the Overview.csv Resource File***.
`resources\Profiler\Definition`	Resources used by the MashApp. 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. See also ***Editing the Overview.csv Resource File***.
`assets\colorschemes`	Color schemes. 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. If the data file resides in a subdirectory of `resources\Profiler`, specify the relative path and the file name. For example: 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. 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.

手順 73.3. 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.

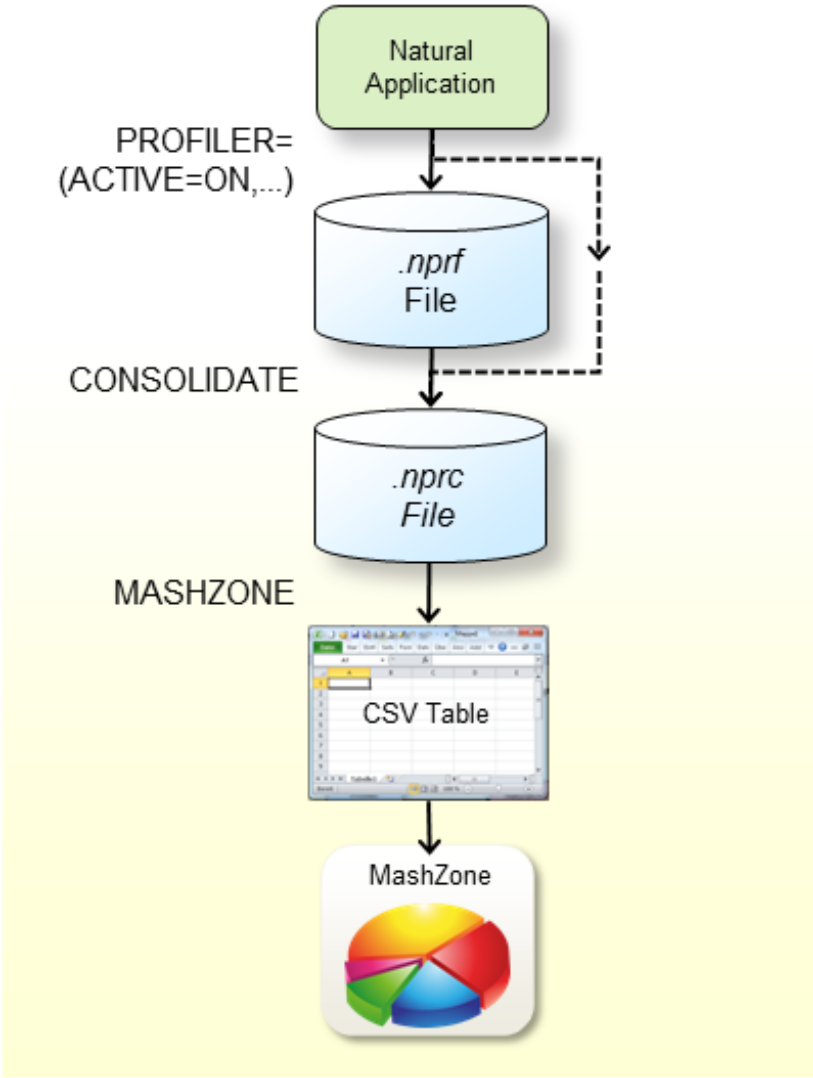
   The MashApps in the `importexport\Profiler_date` directory are named as follows:

`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 session by switching on the `ACTIVE` subparameter of the `PROFILER` profile parameter in the NATPARM parameter file or dynamically when invoking Natural. See *PROFILER – Profile a Natural Session* in the *Parameter Reference* documentation. 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 `EVENTTRACE=OFF` with the `PROFILER` profile parameter when you profile the Natural 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. Use as Work File 7 a CSV (comma-separated values) file in the Natural Profiler data directory (see ***Unpacking the Zip File***) in the MashZone environment.
- 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.

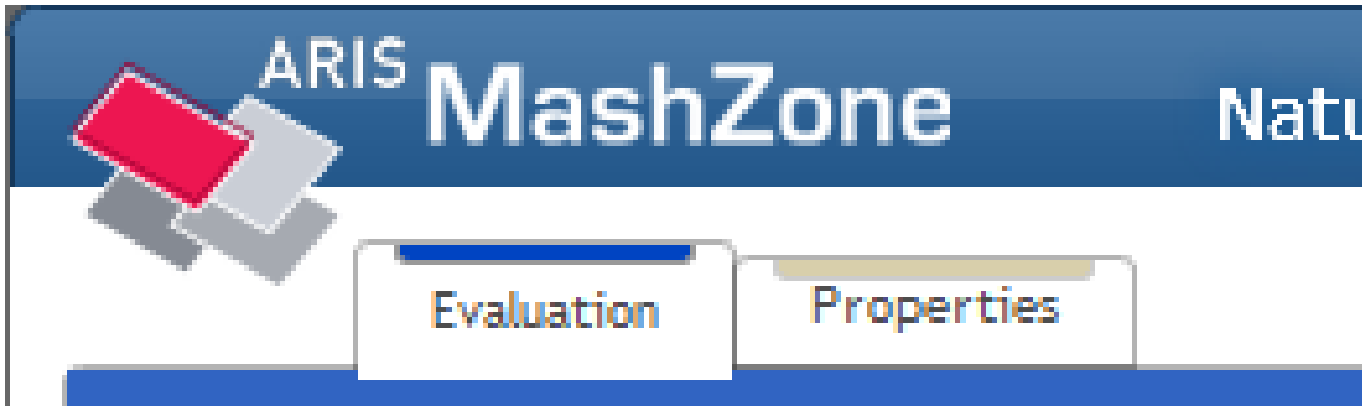
If you already started the Natural Profiler MashApp earlier, MashZone may not immediately detect the new entry in the `Overview.csv` file. In this case, start any other MashApp, and then restart the Natural Profiler MashApp to clear the internal MashZone buffer.

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.

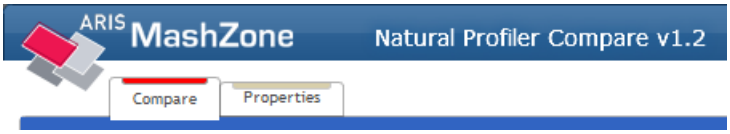
   The Natural Profiler MashApp offers two tabbed pages for analyzing the Profiler event data and viewing the Profiler properties and statistics:



   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:



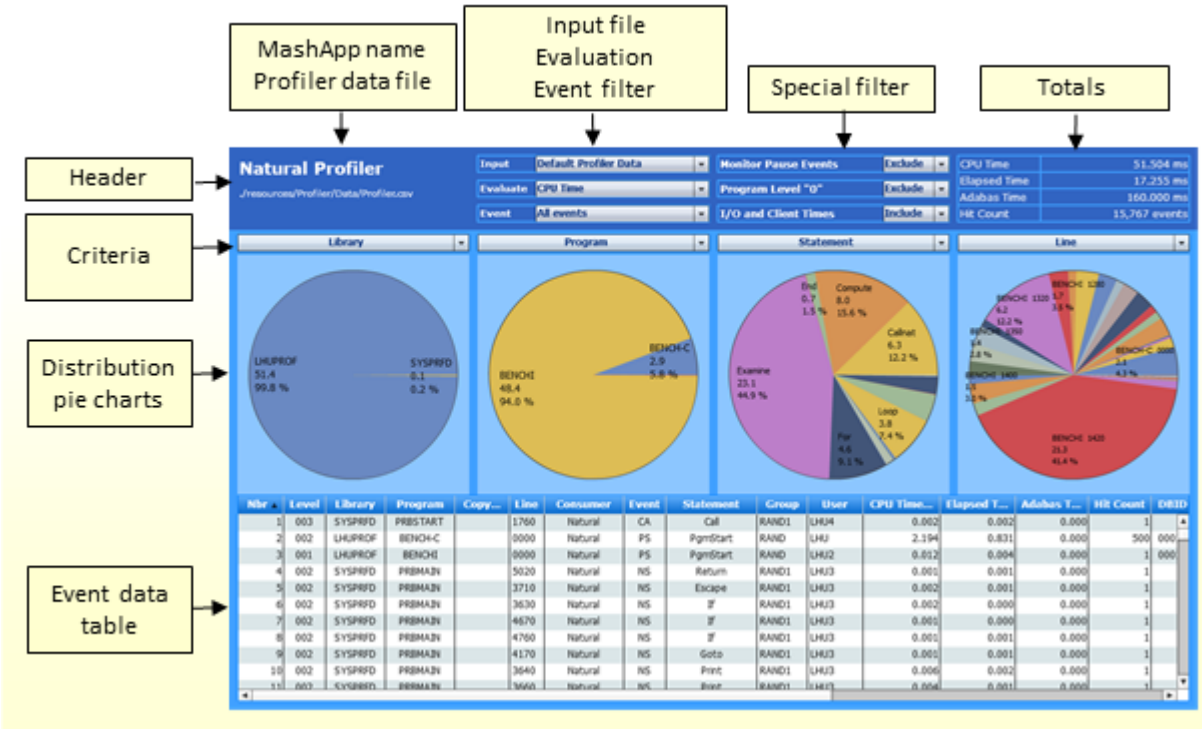
The **Compare page** compares the Profiler event data of two monitored applications.

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. 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). On UNIX and Windows, the entire operation is charged to the **Administration** consumer.

Consumer	Event Type	Description
Database	DB	The time consumed for database calls. 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. When you analyze the elapsed time of an interactive application, this section shows the user response time. 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. 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. When you analyze the elapsed time of an interactive RPC application, this section shows the user's response time. 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*.

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.

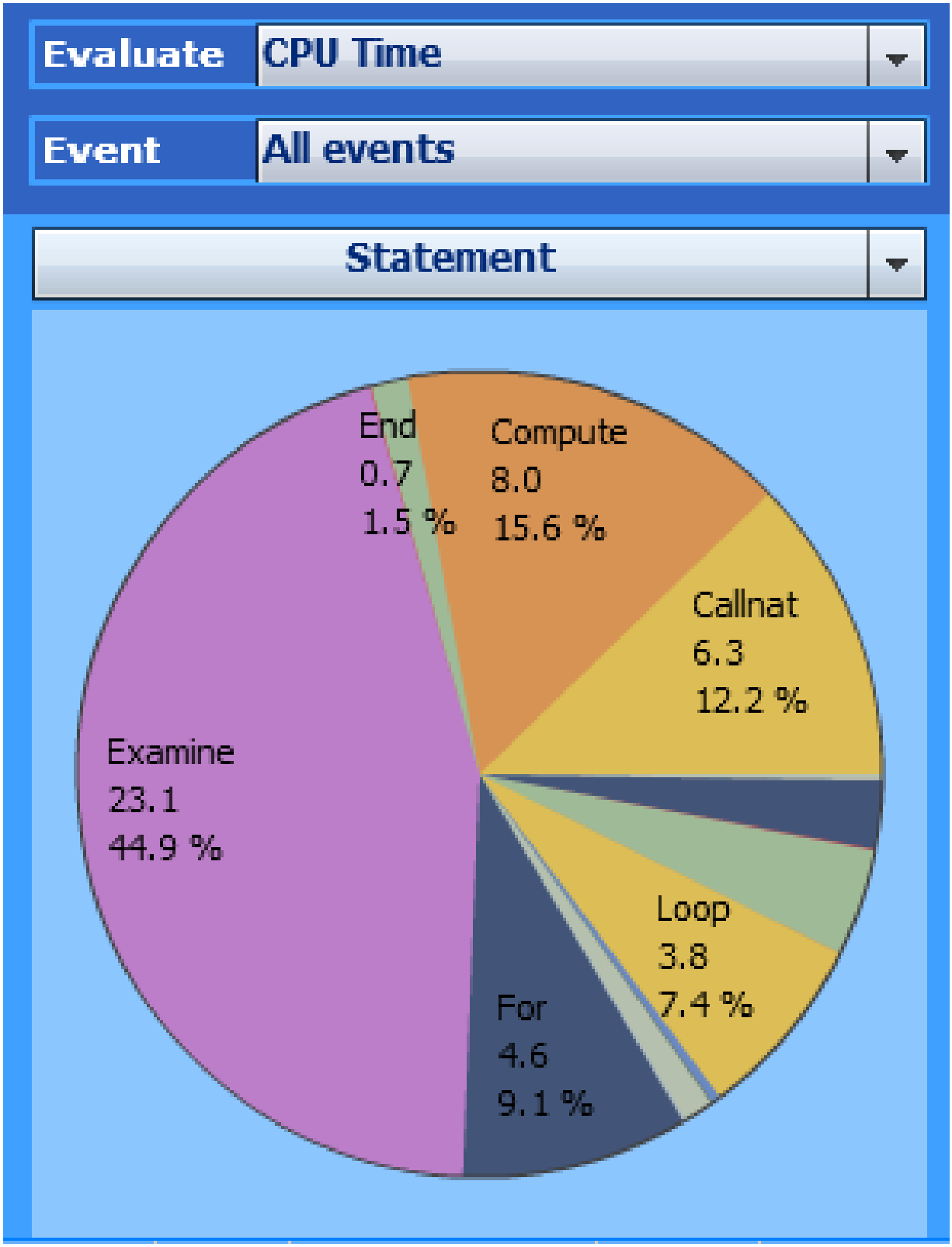
Type

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

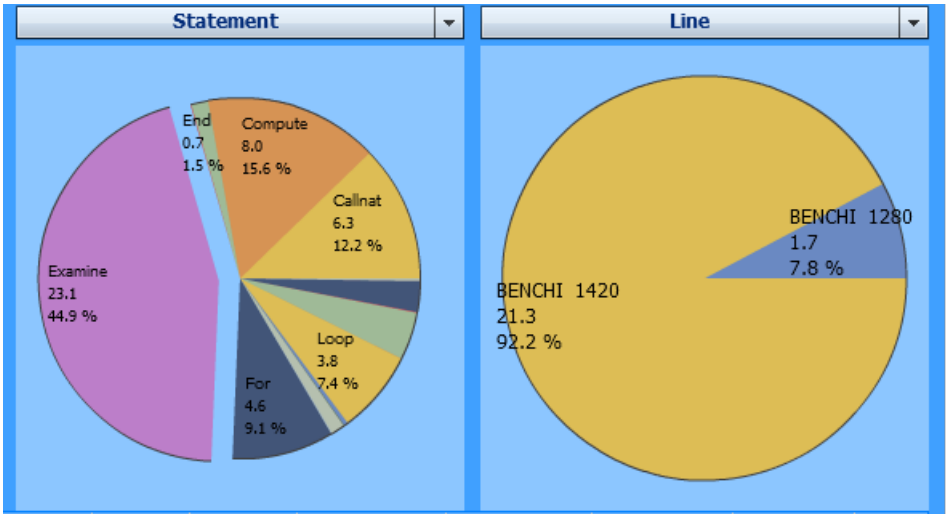
This section describes how you can evaluate distribution pie charts.

- 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.

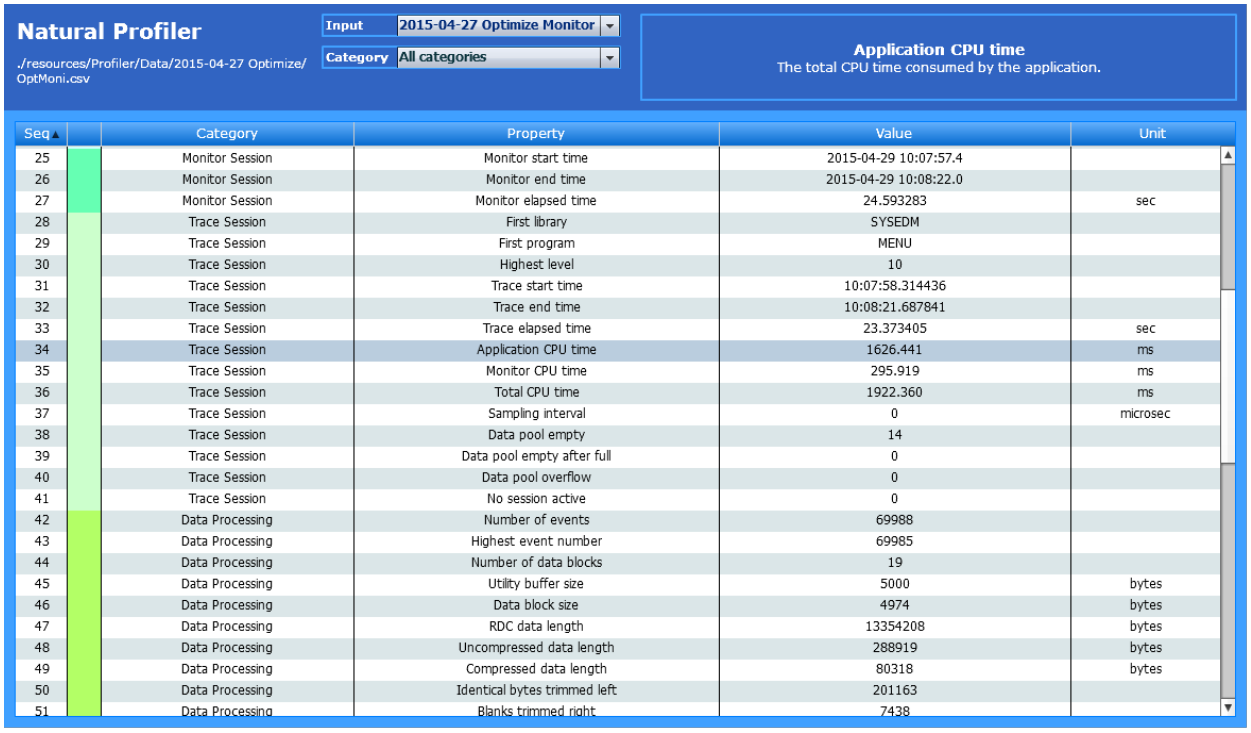
Compare Column Chart

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

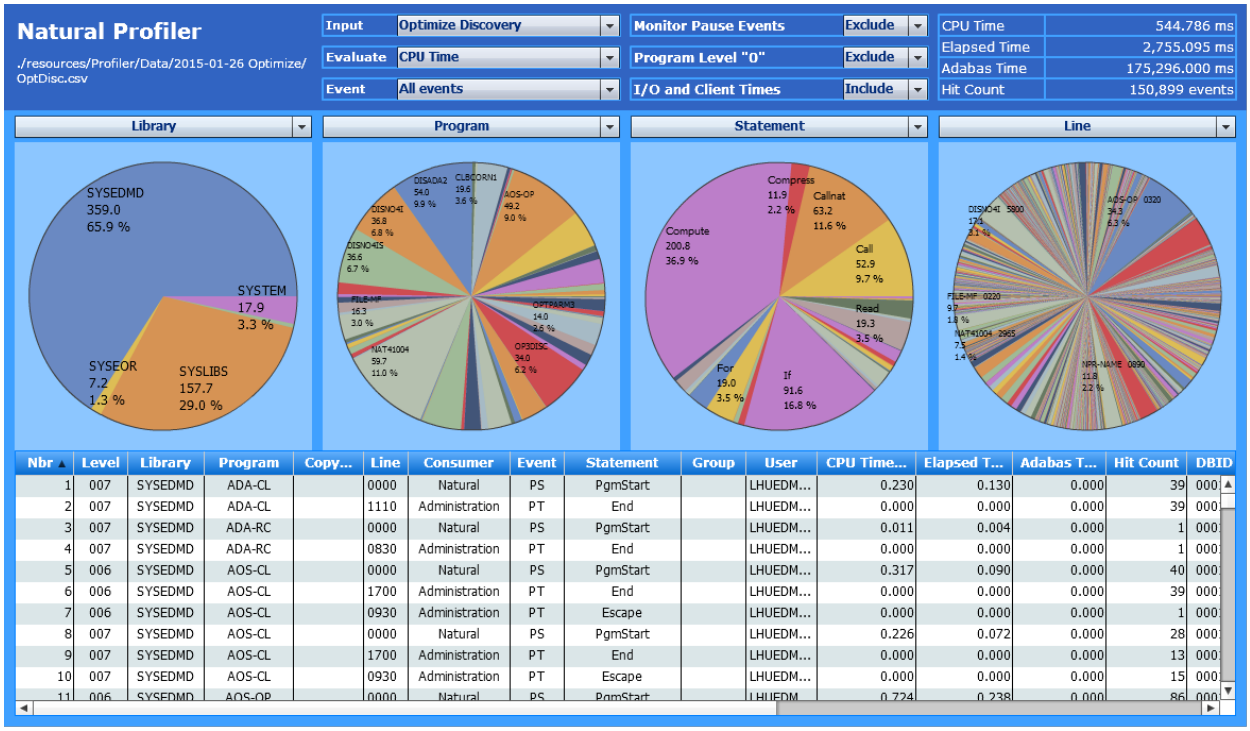
This section describes the following use cases:

- 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

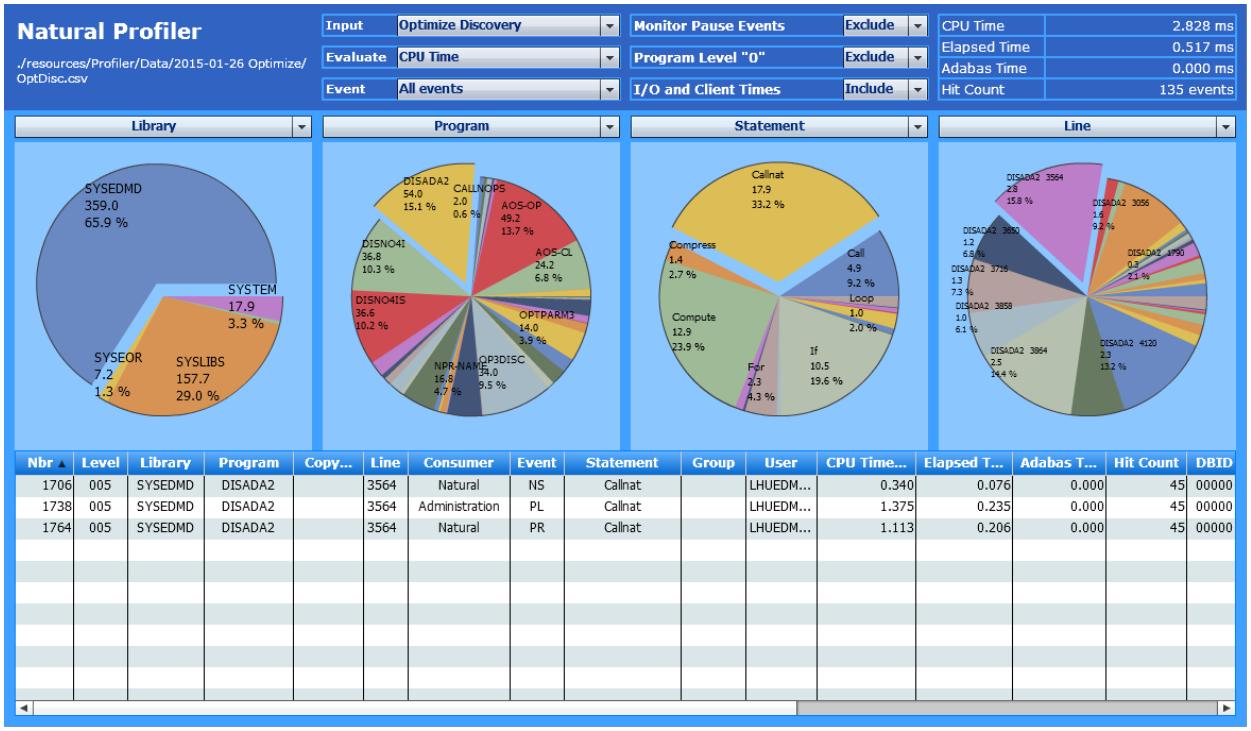
Event: All events

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:

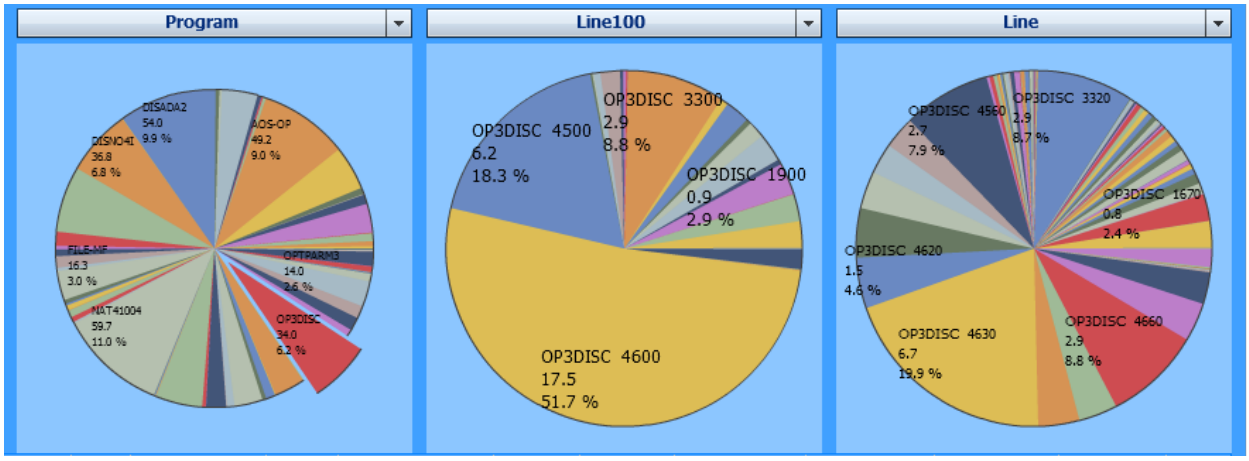


The event data table and the total in the page header now only refer to Line 3564 where the program executes the `CALLNAT` statement. The `CALLNAT` statement caused the event types (`NS`, `PL` and `PR`), each executing `45` times which results in a total **Hit Count** of `135` events.

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.

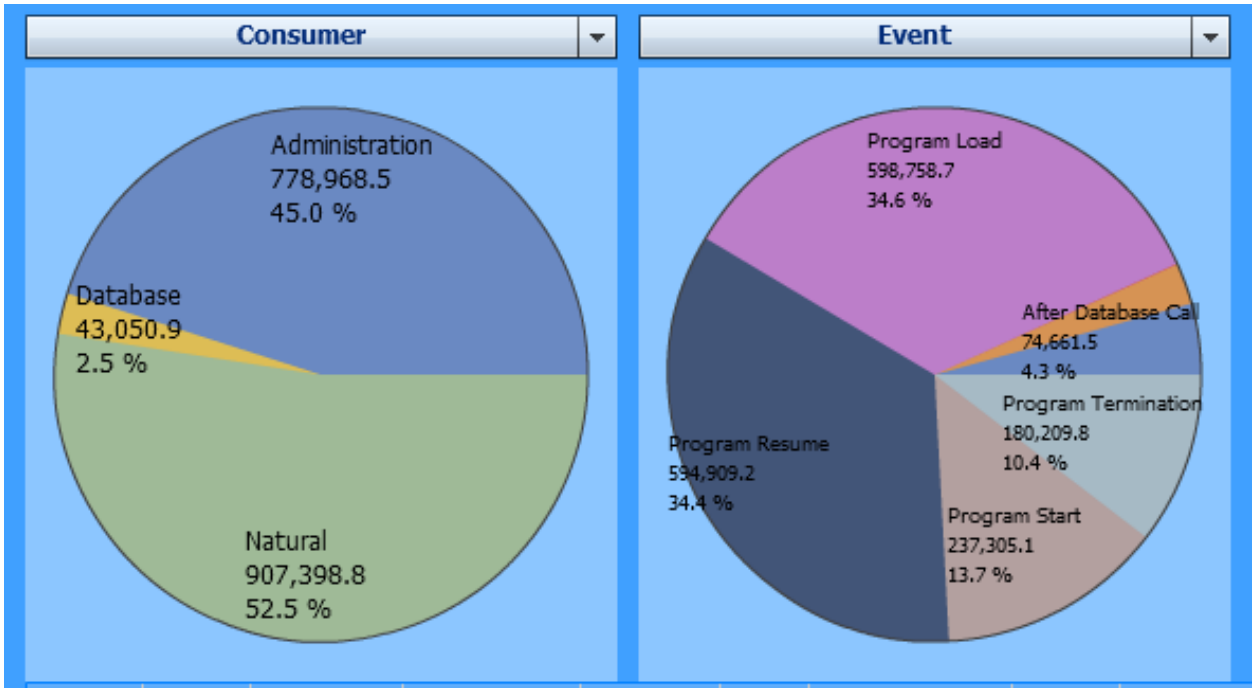
The **Line100** chart shows that more than half of the time was consumed by the statements in the lines ranging from 4600 through 4690. Additionally, considering the statements in the lines ranging from 4500 through 4590, this part of the program even consumes 70 percent of the entire execution time. 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

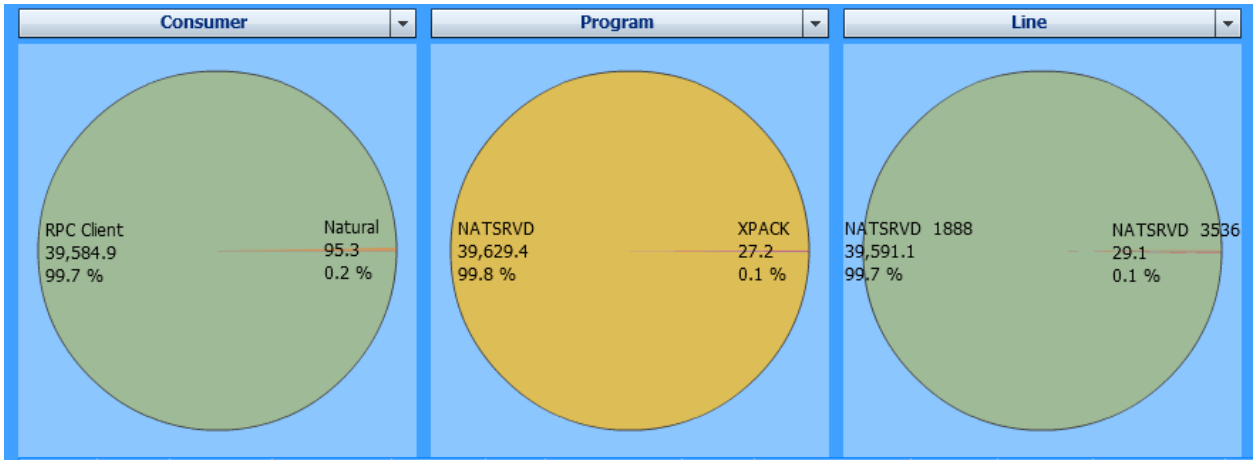
Event: All events

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.

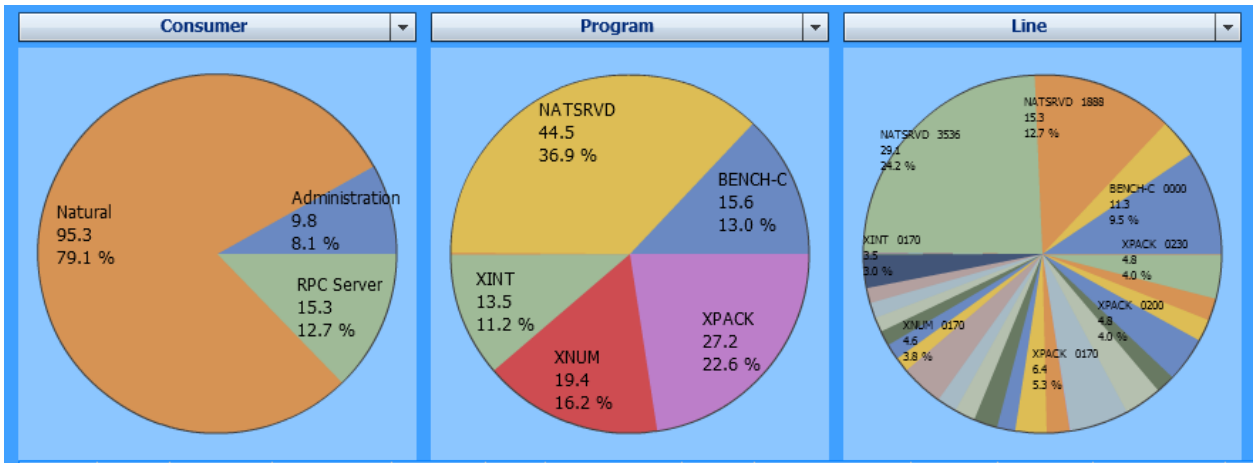
In the example below, the Natural RPC client consumes nearly all of the elapsed time:



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

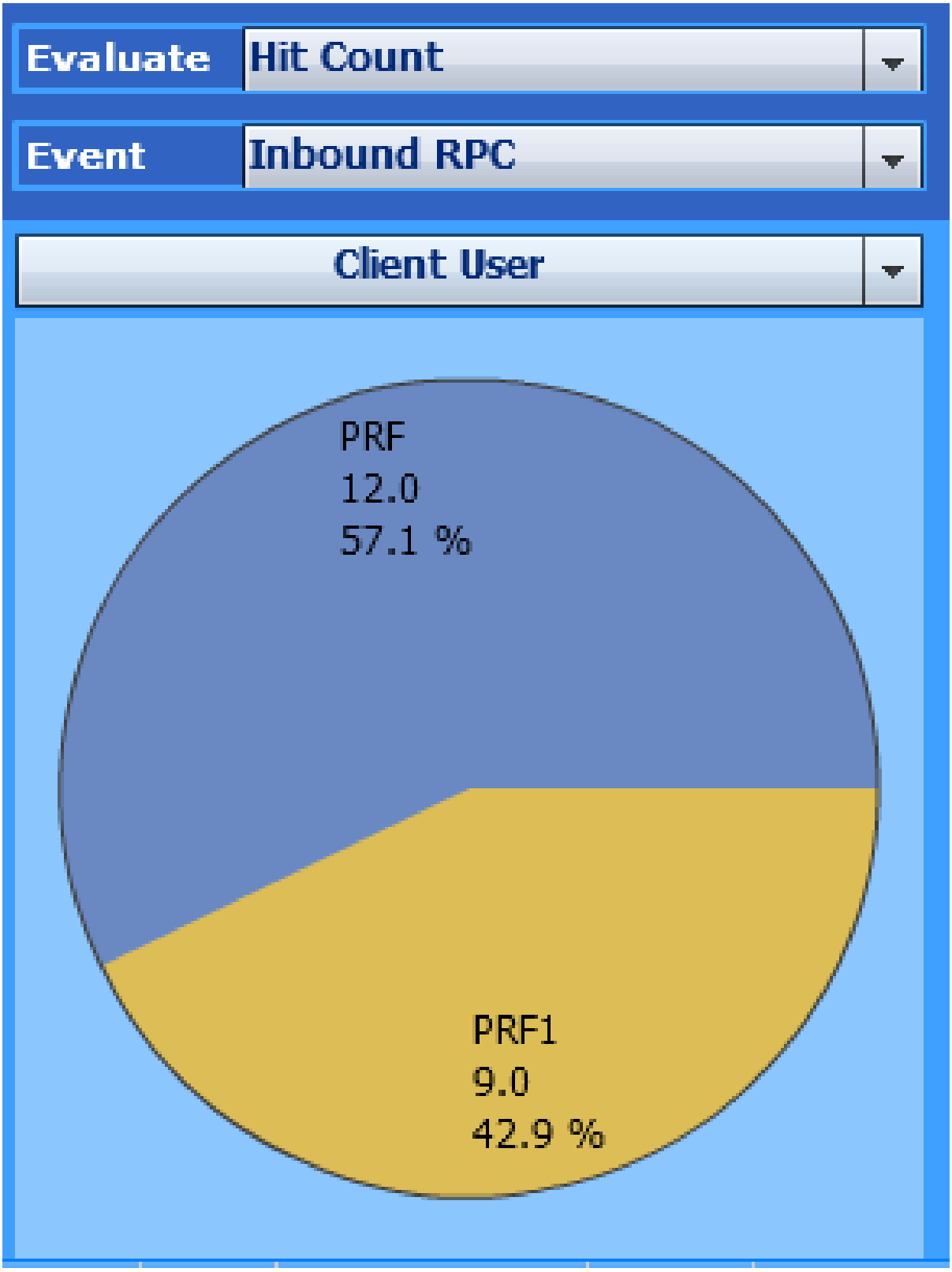
Criteria: Consumer, Program, Line

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:

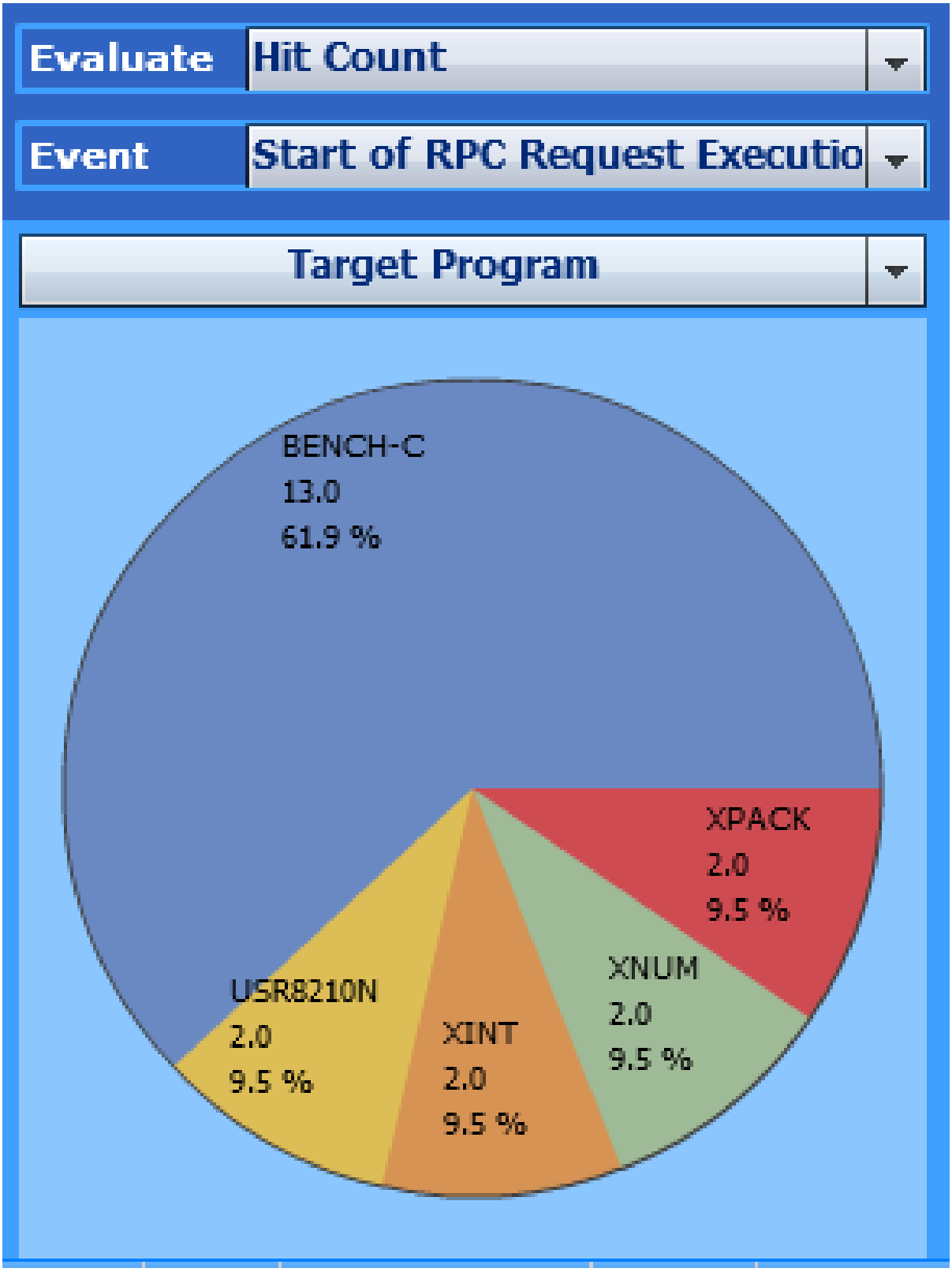
Evaluate: Hit Count

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

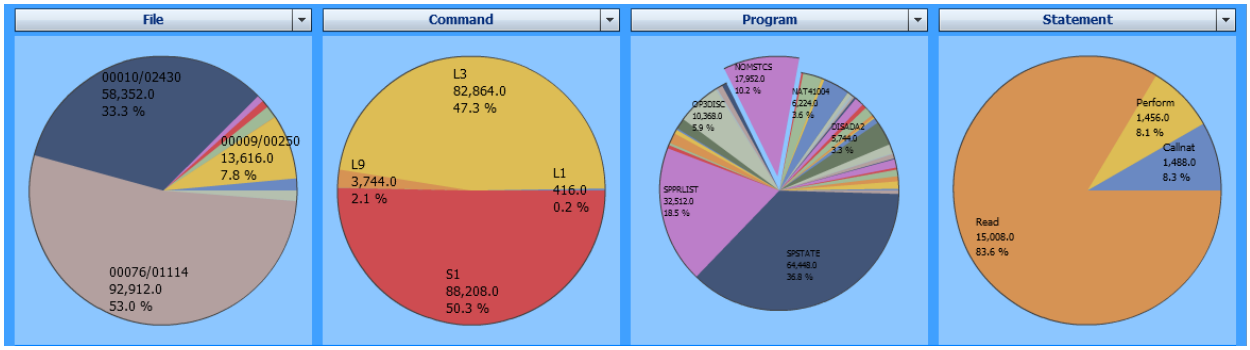
Criterion: Target Program

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

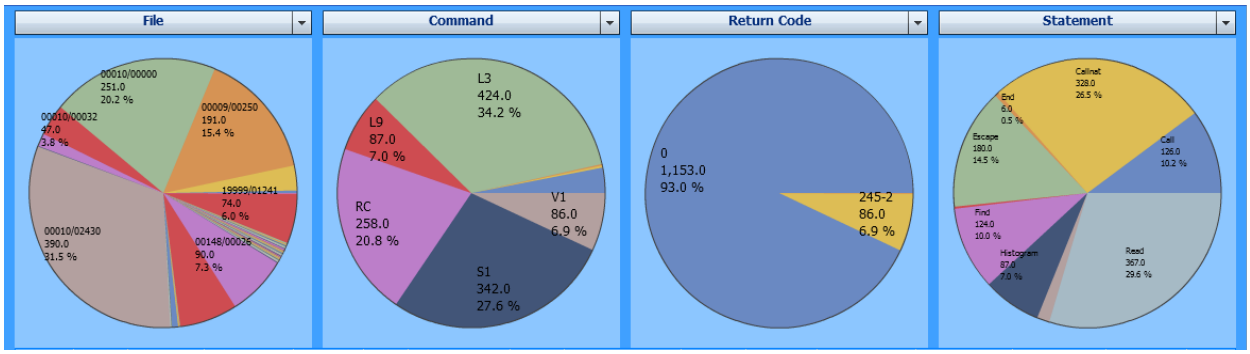
Event: All events

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.

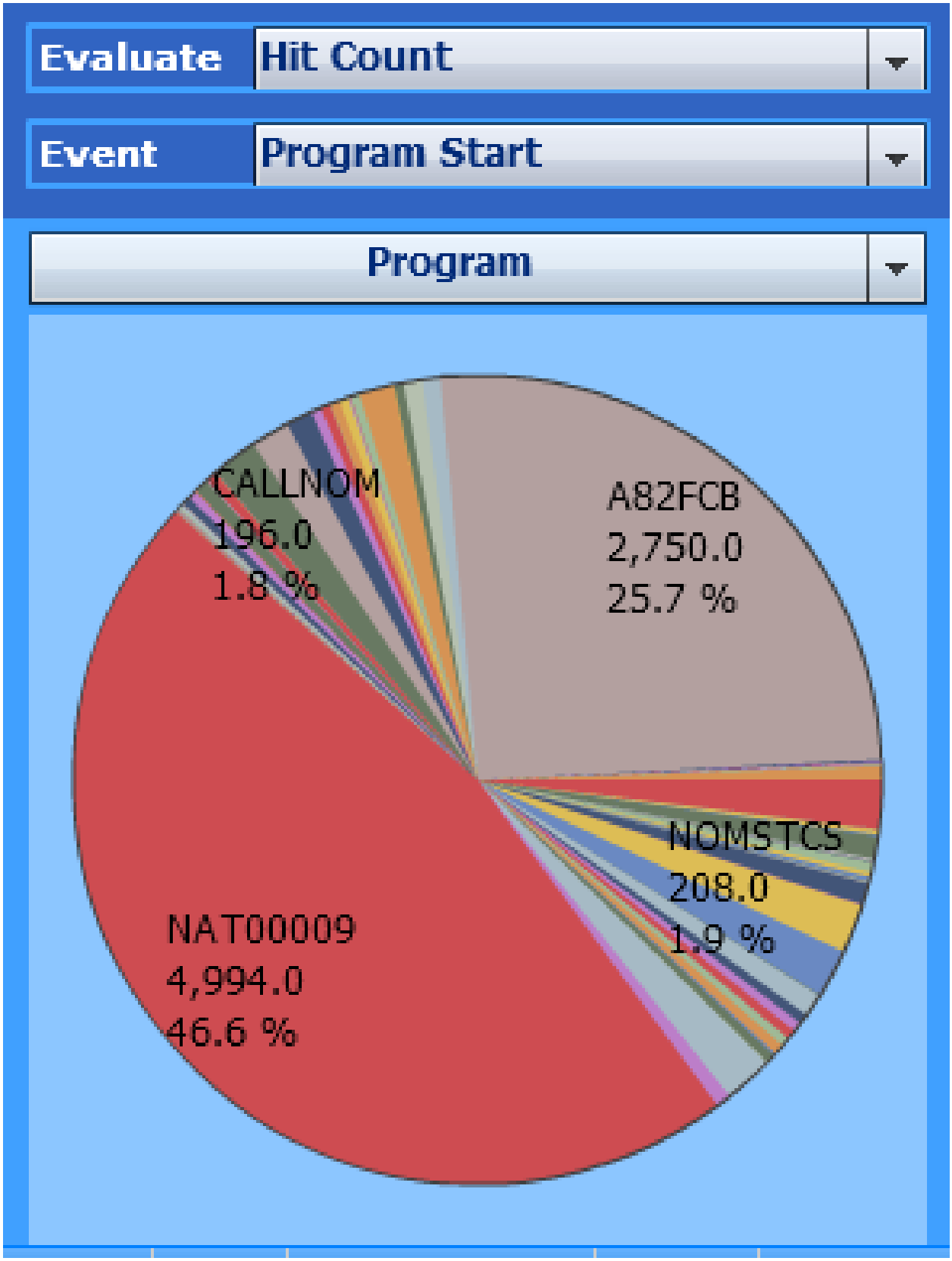
- How often were Natural objects started?
- How many statements were executed in the monitored programs?
- Which objects were called by a selected program and how often?
- How often was a selected object called?

- 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

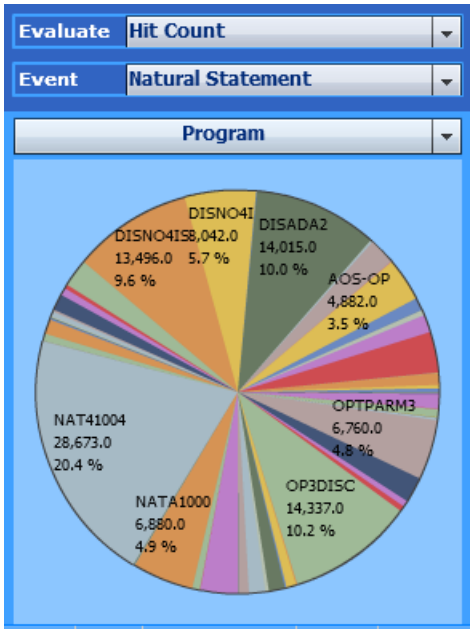

Evaluate
Hit Count
Event
Program Start
Program
CALLNOM
196.0
1.8 %
A82FCB
2,750.0
25.7 %
NOMSTCS
208.0
1.9 %
NAT00009
4,994.0
46.6 %

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

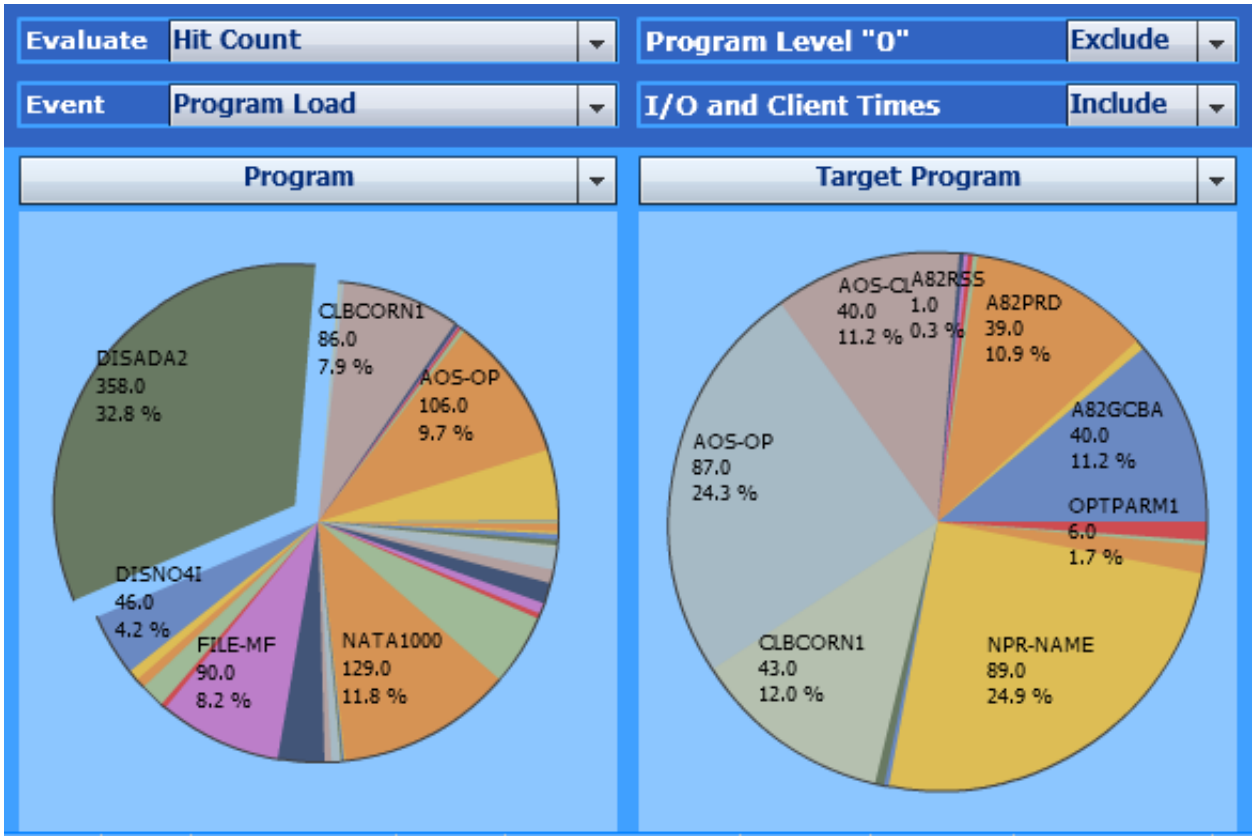


In the example above, the program `NAT41004` executed `28,673` statement events.

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

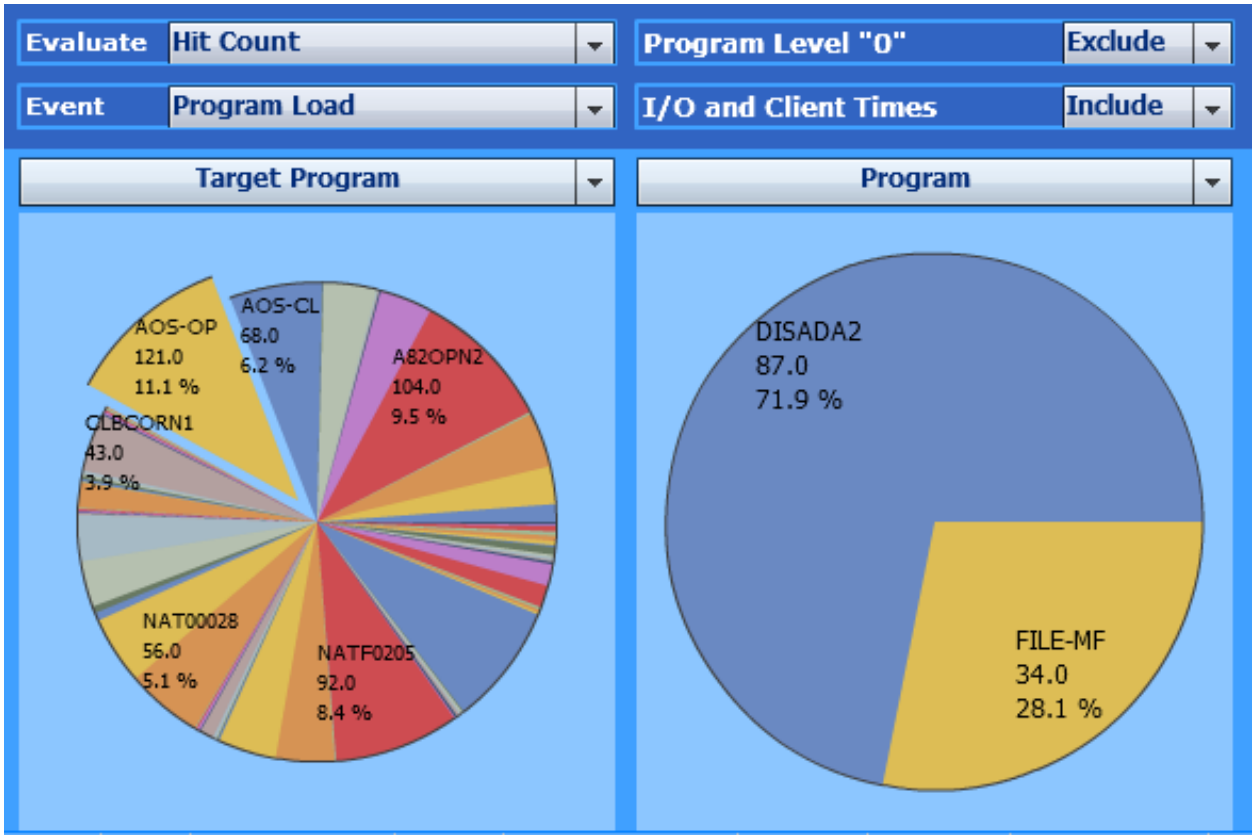


In the example above, the right pie chart shows the Natural objects called by `DISADA2`. The Natural object `NPR-NAME` was called `89` times.

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`.

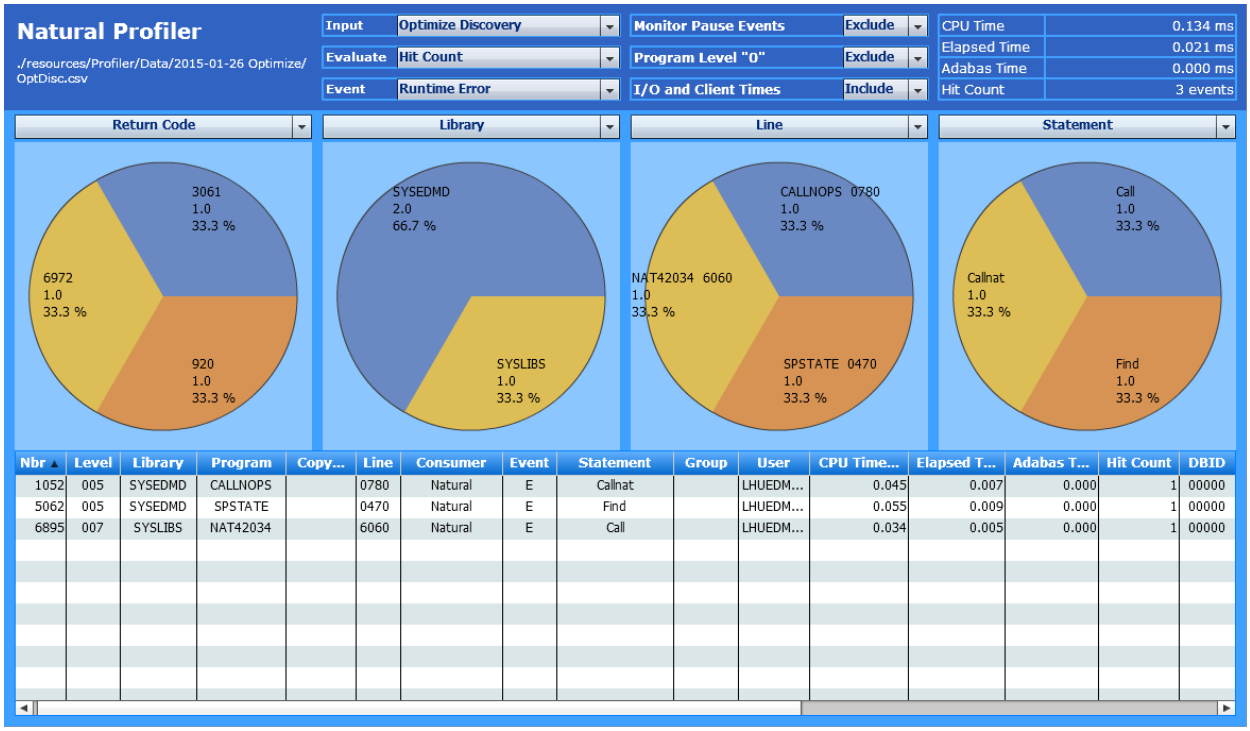
How many runtime errors occurred?

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.

XIII 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

Remote Directory Maintenance

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.

74 Invoking and Terminating SYSRPC

- Invoking SYSRPC 580
- Terminating SYSRPC 580
- Invoking Online Help 581

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 a system command.

手順 74.1. To invoke SYSRPC

- In the Command line, enter the following command:

  ```
  SYSRPC
  ```

  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

手順 74.2. To terminate the SYSRPC utility

- In the **Code** field of the **Client Maintenance** menu, enter a period (.).

  または:

  Choose PF3 (Exit).

Invoking Online Help

手順 74.3. To invoke the online help function

- Choose PF1 (Help).

75 Service Directory Maintenance

- Service Directory Concept 584
- Invoking Service Directory Maintenance 585
- Fields on the Service Directory Screen 587
- Commands for Service Directory Maintenance 590

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 profile parameter `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 profile parameter `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 profile parameter `SRVNODE` 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.

手順 75.1. 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. The maximum length of input is as follows: Standard view of the **Service Directory** screen: 16 characters 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: 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. The maximum length of input is as follows: Standard view of the **Service Directory** screen: 16 characters Extended node/server view of the **Service Directory** screen: 32 characters
Logon	Initiates a Natural logon to the server. 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. Possible values are as follows: 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*). N or *blank*: If set to N (No) or if no value is entered, no logon is initiated. 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. 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, 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. 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.

手順 75.2. 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. With the direct command `EXPIRATION`, you can enter an expiration time in seconds, for example, `EXPIRATION 86400`. Maximum is an 8-digit number. 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: **Find what** — Enter an alphanumeric search string of up to 32 characters. **Case sensitive** — Replace the default setting `N` (No) by `Y` (Yes) to distinguish between uppercase and lowercase characters. **Whole words only** — Replace the default setting `N` (No) by `Y` (Yes) to search for whole words only. Choose ENTER to start searching and move from one hit to the next if one exists. Press PF4 to restart searching from the beginning. 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): **Find** — Enter an alphanumeric search string of up to 32 characters. **Replace with** — Enter an alphanumeric replace string of up to 32 characters. **Whole words only** — Replace the default setting `N` (No) by `Y` (Yes) to search for whole words only. **Search only** — All names in the service directory are searched for matches by default (blank field entry). You can enter one of the following letters to restrict the search to one of the following items: `N` — Node names only `S` — Server names only

Direct Command	PF Key	Function
		L — Library names only P — Program names only **Replace all** — Replaces all occurrences of the search string found. Choose ENTER to start searching and move from one hit to the next if one exists. Press PF4 to restart searching from the beginning. 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.

76 Replacing Items in the Service Directory

- Syntax of SYSRPC SM REPLACE 596

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. 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. This is the default value.
WHOLE	Replaces only occurrences that match the whole search string.

注意: The search operation is not case-sensitive.

77 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.

注意: The subprogram 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**

78 Generating Single Interface Objects with Parameter Specification

- Using the Interface Object Generation Function 602
- Specifying Parameters 604
- Examples of Interface Object Generation 607

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.

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`.

重要: 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.

手順 78.1. To generate a single interface object

1 Before you invoke the SYSRPC utility, log on to the library into which you want to generate the interface object.

2 In the **Code** field of the **Client Maintenance** menu, enter the following command:

```
IG
```

3 Choose ENTER.

The **Generate Client Stub Routine** window 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 current library and cannot be changed.

DBID, FNR are non-modifiable fields that display the database ID (DBID), the file number (FNR) and the type of Natural file (`FNAT` = system, `FUSER` = user) for the current library.

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 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, the parameter definitions of the respective subprogram are displayed on the **Interface Object Generation** screen.

If the specified name is identical to an 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 name entered in the **Program Name** field, does *not* correspond to 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 | NONE | void
	IDL Generate an interface object according to IDL requirements. NONE Generate an interface object according to Natural requirements. void Show COMPAT setting. **注意:** See also: *Special Considerations for Reliable RPC* and *Special Considerations for Calling EntireX RPC Servers* .
LIMIT	32000 | 1GB | void
	32000 Sets the upper size limit to 32000 bytes. 1GB Sets the upper size limit to 1 GB. 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 in the assigned library.

The **SYSRPC - Information** window appears which indicates the size 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 generating the interface object.

If the interface object was generated in the Natural system library SYSRPC, you it object 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. A level can be a number in the range from `01` (highest level) to `99` (lowest level). The leading `0` is optional. See also *Defining Groups* and *Example 2* for an example of a group definition.
Attr	The attribute of the parameter: `M` (modifiable - INOUT), `O` (output - OUT) or `I` (input - IN). 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*. If an interface object has been generated from a subprogram, the attribute is `M` by default, which may need modification. If an interface object has been generated from another interface object, the attribute values specified for the original object are retained. 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. 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`. This field does not apply to the following Natural data formats: `D` (date), `G` (group), `L` (logical) and `T` (time). The Natural data format `A` is restricted to 1073739357 bytes, Natural data format `B` is restricted to 536869678 bytes. `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. The precision of the parameter, that is, the number of digits after the decimal point.
Dimension 1/2/3	Only applies to arrays. Optional. The first, second and third dimension of the parameter. An X-array or an X-group array is specified by entering an asterisk (*) for a dimension. 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.

注意: Any group definitions in a subprogram will be ignored when an interface object is generated from this subprogram. In this case, you have to define the group again on the **Interface Object Generation** screen and adapt the dimension of the group elements accordingly. (Dimensions defined within a group are propagated to the parameter definitions at a lower level.) If you generate an interface object from another interface object that contains a group, the group definitions will be retained.

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:*)`.

注意: If you generate an interface object from a subprogram 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				

79 Generating Multiple Interface Objects

- Using the SYSRPC SGMASS Direct Command 612
- Name Specification and Compression 614

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.

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) 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.

注意: If you do not specify an object name or a range of names, with few exceptions (see below), all subprograms 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. This is the default setting.
value	A subprogram with a name equal to *value*.
*value**	All subprograms with names that start with *value*.
value<	All subprograms with names less than or equal to *value*.
value>	All subprograms 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.

80 Calculating Size Requirements

- Using the SYSRPC CSMASS Direct Command 618
- Name Specification and Compression 620

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) 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 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. This is the default setting.
value	A subprogram with a name equal to *value*.
*value**	All subprograms with names that start with *value*.
value<	All subprograms with names less than or equal to *value*
value>	All subprograms 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.

81 Parameter Maintenance

- Invoking Parameter Maintenance 622
- Specifying Profile Parameters 622

Applies to client sessions only.

The **Parameter Maintenance** function is used to dynamically (within a session) modify RPC-specific Natural profile parameters.

注意: 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

手順 81.1. 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 Profile Parameters*.

3 Choose PF3 (Exit) to save any modifications and exit the **Client Parameter Maintenance** screen.

または:

Choose PF12 (Canc) to exit without saving any parameter modifications.

The **Client Maintenance** menu appears.

Specifying Profile Parameters

In the input fields provided on the **Client Parameter Maintenance** screen, you can modify the settings of the profile parameters described in the table below:

Field	Explanation
Timeout	Specifies the number of seconds the client is to wait for an RPC server response. See also the profile parameter `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 profile parameter `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. See also the profile parameter `COMPR` described in the *Parameter Reference* documentation. 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 the section *Profile Parameters* in the *Parameter Reference* documentation.

82 Server Command Execution

- Using Server Command Execution 626
- Pinging an RPC Server 628
- Terminating an RPC Server 630

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 profile parameter `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

手順 82.1. 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. **Server** node: Pings the selected RPC server. 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. 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**.

手順 82.2. 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.

または:

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 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 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. 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.

手順 82.3. 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.

手順 82.4. 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.

83 Listing Servers Registered on EntireX Broker

- Example of an SYSRPC SRVLIST Direct Command 635
- Viewing a Server List 635
- Viewing Additional Server Information 636
- Customizing Server Lists 637

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.

注意: 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 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 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. 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. 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. 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.

手順 83.1. 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 profile parameter `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.

84 Remote Directory Maintenance

- Using Remote Directory Maintenance 640
- Fields on the Editor Screen of a Remote Directory 641
- Commands for Remote Directory Maintenance 642

The **Remote Directory Maintenance** function is used to maintain a remote directory in order to connect the client's calling program to a subprogram on a server.

For further information on how to apply the **Remote Directory Maintenance** function, refer to *Specifying RPC Server Addresses* (*Operating a Natural RPC Environment*), *Using a Remote Directory Server (RDS)* and *Natural RPC Terminology* described in the *Natural Remote Procedure Call (RPC)* documentation.

Using Remote Directory Maintenance

注意: If you create a new remote directory by entering code `C` (see the instructions below), the entries of an existing directory will be overwritten.

手順 84.1. To use the Remote Directory Maintenance function

1 In the **Code** field of the **Client Maintenance** menu, enter the following command:

```
RD
```

2 Choose ENTER.

A window appears.

3 In the input field, enter either of the following commands:

```
C
```

to create a directory or

```
M
```

to modify a directory.

4 Choose ENTER.

An additional window appears.

5 Enter an expiration time in seconds (see also ***Expiration Time*** below) and choose ENTER.

An editor screen similar to the example below appears:

```
-----------------------------            S 01- ---------------Columns 001 072
 ====>                                                        SCROLL===>  CSR
****** NODE      L T    SERVER    L T    LIBRARY   L T    PROGRAM   L T
****** ****************************** top of data ******************************
000001 ETB01     Y B    NRPC2301         SYSTEM           SUB1
000002                                                    SUB2
000003                  NRCP2301         SYSTEM           SUB3
000004                  NRPC2302         SYSTEM           SUB4
000005                                                    SUB5
000006                                                    SUB6
000007 ETB01            NRPC2301  Y      SYSTEM           SUB7
000008                                                    SUB1      Y
000009                                                    SUB2
000010                  NRCP2301         SYSTEM           SUB3
000011                  NRPC2302         SYSTEM           SUB4
000012                                                    SUB5
000013 ETB01     Y      NRPC2301         SYSTEM           SUB6
000014                                                    SUB1      Y
000015                                                    SUB2

000016 ETB01     Y      NRPC2301         SYSTEM           SUB3
Enter-PF1---PF2---PF3---PF4---PF5---PF6---PF7---PF8---PF9---PF10--PF11--PF12---
      Help        Quit  Save  Find  Chng  Up    Down                    Canc
```

Expiration Time

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 expiration time is set to `0`, the remote directory data will not be reloaded.

注意: If you create a new directory (code `C`), you invoke an input line by entering the line command `I` at the beginning of the line `top of data`. See also *Line Commands* below.

Fields on the Editor Screen of a Remote Directory

The fields contained in the editor screen of **Remote Directory Service** maintenance are identical to the fields described in *Fields on the Service Directory Screen* in the section *Service Directory Maintenance*. The field **L** is the equivalent to the field **Logon**.

In addition to the fields provided on the **Service Directory Maintenance** screen, the field **T** (Transport) appears: enter a `B` for EntireX Broker.

Commands for Remote Directory Maintenance

This section contains information on the commands provided on the editor screen of **Remote Directory Service** maintenance.

- Line Commands
- Direct Commands and PF Keys

Line Commands

The line commands available on the editor 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 line(s) marked with `C`, `CC`, `M` or `MM` below the line in which the command was entered.
`B`	Copies or moves the line(s) marked with `C`, `CC`, `M` or `MM` above the line in which the command was entered.
`C`	Marks the single line to be copied.
`CC`	Marks the block of lines to be copied.
`D(n)`	Deletes one or *n* lines beginning with the line in which the command was entered. *n* can be in the range from 1 to 9.
`DD`	Marks and deletes a block of lines. Mark the block of lines by entering the command in the first and last line of the block, and choose ENTER to execute the command.
`I(n)`	Inserts one or *n* empty lines below the line in which the command was entered. *n* can be in the range from 1 to 9.
`M`	Moves a single line below the line in which the command was entered.
`MM`	Marks the block of lines to be moved.

手順 84.2. To copy or move a block of lines

1 At the beginning of the line where the block starts, enter 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, enter either of the following line commands:

```
CC
```

to copy the block or

```
MM
```

to move the block.

3 Choose ENTER.

The block of lines is marked which is indicated by the message: `Block is pending.`

4 At the beginning of the line below or above which you want to place the block, enter either of the following line commands:

```
A
```

to copy or move the block *below* the specified line or

```
B
```

to copy or move the block *above* the specified line.

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 and PF keys are provided on the editor screen:

Direct Command	PF Key	Function
`RESET`		Removes the line marks set with a line command (see ***Line Commands***) or with the direct command `CHANGE` (see below).
`TOP`		Scrolls to the beginning of the list.
`BOT`		Scrolls to the end of the list.
`FIND` *`string`*		Scans the editor for a *`string`* of characters, for example: `FIND ETB1`. Choose PF5 (Find) to scan for the next occurrence.
`CHANGE` *`string1 string2`*		Replaces character *`string1`* by *`string2`*, for example: `CHANGE ETB1 ETB2`. Choose PF6 (Chng) to replace the next occurrence.
	PF1	Help. Invokes the online help.
	PF3	Quit. Saves any modifications and exits the editor screen.

Direct Command	PF Key	Function
	PF4	Saves any modifications.
	PF5	Find. Scans for the next occurrence of the character string specified with the direct command `FIND` (see above).
	PF6	Change. Replaces the next occurrence of the character string specified with the direct command `CHANGE` (see above).
	PF7	Up. Scrolls one page backward.
	PF8	Down. Scrolls one page forward.
	PF12	Cancel. Exits the editor screen without saving modifications.

85 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 ]
```